Appendix F

Abstraction Based Complexity Management

F.1 Overview

The overall goal of the Abstraction Based Complexity Management seedling was to research existing drivers of complexity in aerospace systems, and notionalize a new design paradigm that could significantly reduce the cost and schedule associated with creating these systems. The fundamental areas of research considered as part of this activity included

- Develop a measure of complexity that utilizes available systems parameters.
- Assess the uncertainty of complex hybrid systems and the relationship to complexity.
- Define an abstraction-based design method to provide formalism to the definition of the system.
- Determine a method of architecture synthesis that can be used to explore the complete design space available during early conceptual design.
- Assess analytical methods available to identify weakly connected areas in a complex system for the purpose of clustering.

Each of these topics is discussed in detail in the the following sections of this Appendix. Each section comprises a draft of a stand-alone paper with the intention that each will be presented at an appropriate future industry conference. A brief discussion of the major topics of the seedling activity is provided here for context.

Complexity Measure

To enable a design process that utilizes complexity as a primary driver, an appropriate definition of complexity is clearly needed. There are many options for defining this quantity, some more useful than others. To be useful, the resultant equation should consist of parameters that are readily available to the designer. While many previous works have considered part count and/or interconnects as an indicator of complexity, this significantly oversimplifies the problem. It is of course possible to design a system that consists of many interconnects, and yet is not complex. Conversely, a system that performs highly complex functions, perhaps with the aid of software, may have relatively few components or interconnects. Also of importance is the nature of the substance that is passed from component to component, particularly in an air vehicle. For example, hydraulic interconnects are typically more complex that pneumatic interconnects due to the nature of the respective fluids and the pressures at which they operate.

In addition to physical interconnects and components, the functional behavior and interaction between components and subsystems is another element of complexity. The potential for dynamic instabilities increases as systems become more interconnected and time scales become increasingly separated. While a single, robust method for capturing these behaviors is still under development, a general framework is defined here.

$$C\left(n,A\right)=\sum_{i=1}^{n}\alpha_i+\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{4}\beta_k\alpha_{ijk}+\gamma\left[\frac{\log n}{\log 7}\right]E\left(A\right) \tag{F.1}$$

The overall representation for complexity developed under this effort is shown in Eq. F.1. The first term represents the number of components in the system under consideration. Note that this is not simply a summation, but a weighted sum that captures inherent complexity associated with an individual component. This is an important element to capture to enable tradeoffs where component reduction is achieved at the expense of fewer, more complex components. An example of this can be seen in the trend toward reduced airfoil count in gas turbines. Turbine designers have been moving toward reduced airfoil counts for years in attempts to reduce cost. However, reducing the number of airfoils in a turbine requires that each airfoil now produce more thrust. In addition, each blade will now be heavier with associated increases in hub stress. Therefore, the design of each airfoil has become more complex in order to meet the desired goal of reduced blade count. This inherent complexity is captured by the α_i terms in Eq. F.1.

The second term in Eq. F.1 represents the number of interconnects in the system under consideration. Again, however, this is not a simple summation. Also reflected is the inherent complexity of each interconnection. As mentioned previously, passing various quantities between parts will have different levels of complexity associated with them. Other elements of each interconnection can be captured in this formulation as well, such as overall length. These properties are important to capture to reflect manufacturing related elements that are important to overall system complexity.

The final term in Eq. F.1 consists of two quantities: one that represents the level of abstraction currently under consideration ("n"), and the other termed "graph energy". The reader will note that the log terms in this representation include a $\log 7$ in the denominator. This value is set based on the assumption that from the highest to lowest level of abstraction there are seven layers. This is based on a heuristic decomposition of typical aerospace vehicles, and may need refinement for other systems under consideration.

The graph energy term $(E(A))$ captures a number of potentially important elements of a system. At a high level, this term represents the density of connections in the system with respect to the total number of possible interconnections. The measure of this quantity can be determined in a number of ways. The one we choose here is to take the summation of the singular values calculated from the adjacency matrix of the system:

$$A = U \sum V^T = \sum_{i}^{N} \sigma_i \left(u_i \cdot v_i\right) \tag{F.2}$$

Further details associated with the measure of complexity and its use to assess system architectures is presented in section F.6 and section F.7.

Overall Design Flow for Management of Complexity

To manage the complexity associated with heterogeneous cyber-physical systems, an abstraction-based and model-based design process is suggested here. The specific abstraction-based method utilized is termed Platform-Based Design. This approach has been used with great success in the computer chip and automotive industries. To drive this process, models of components and other

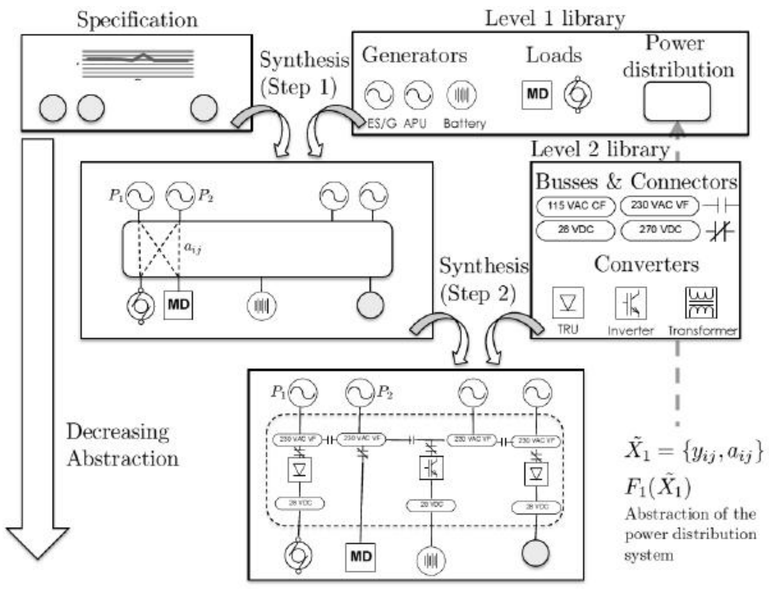


Figure F.1: An example of a platform-based design flow for a cyber-physical system

important aspects of the design (e.g. mission models, cost models, etc.) must be developed. The Platform Based Design method provides formalism to the design process, separating the specification of functionality from that of architecture development, allowing exploration of a larger design space. The system requirements associated with the design are fed into the process as constraints that must be met before the next level of detail can be completed. Architectures are composed from the library of components available to the designer, and the resultant system must provide coverage of the overall requirements space. An overall view of this process is presented in Fig. F.1.

With an architecture thus developed, analysis of the system can be accomplished to peform an assessment of additional important parameters. For example, the graph of the system can be analyzed to look for potential dynamic instabilities, and the design structure matrix can be assessed for problematic interfaces. The overall design flow showing this process is presented in Fig. F.2. Further detail of the overall flow is provided in section F.2. In addition, detail of the Platform Based Design process and its application to an example electric power system is provided in section F.3.

Architectural Enumeration

The synthesis of candidate architectures that meet the requirements set forth in the system specifications is a critical element to allow the management of complexity. There are a number of methods available to accomplish this. One of these methods falls out directly from the Platform Based Design process. The formalism of Platform Based Design allows the description of a system as a set of equations that, when solved, provides the definition of an architecture that is

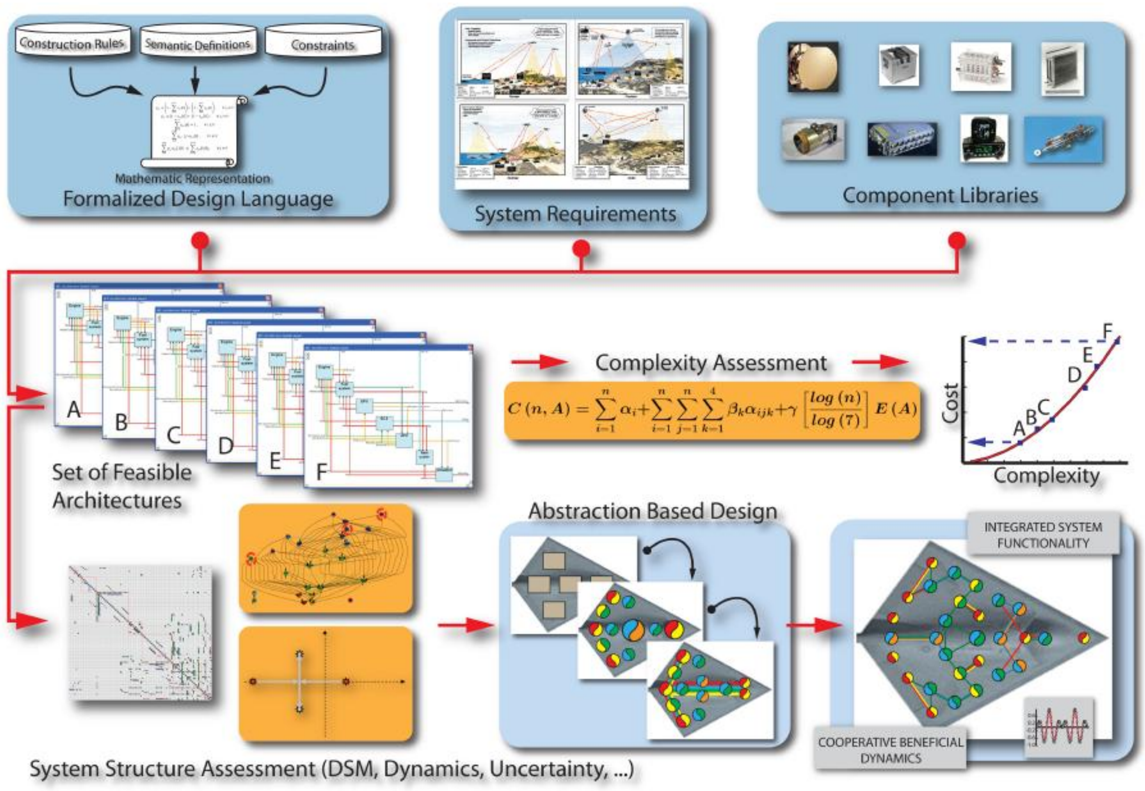

Construction Rules
Semantic Definitions
Constraints
Mathematic Representation
Formalized Design Language
System Requirements
Component Libraries
A
B
C
D
E
F
Set of Feasible Architectures
Complexity Assessment
$C(n,A) = \sum_{i=1}^{n} \alpha_i + \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{4} \beta_k \alpha_{ijk} + \gamma \left[\frac{log(n)}{log(7)}\right] E(A)$
Cost
Complexity
Abstraction Based Design
INTEGRATED SYSTEM FUNCTIONALITY
COOPERATIVE BENEFICIAL DYNAMICS
System Structure Assessment (DSM, Dynamics, Uncertainty, ...)

Figure F.2: Design flow to manage complexity in heterogeneous cyber-physical systems

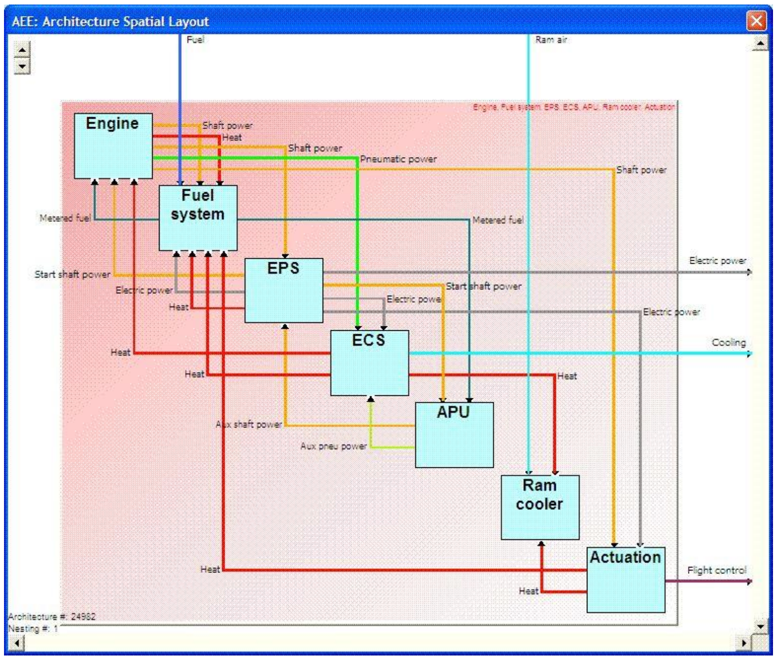


Figure F.3: The ability to synthesize architectures is key to managing complexity

correct-by-construction. An example of this approach is included in the detail of section F.3.

Another approach to developing architectures utilizes a filter-based method that identifies the complete set of possible instantiations, and subsequently identifies the subsets that are feasible, i.e. meet all of the conditions specified for interconnections amongst components. An example of one such architecture is shown in Fig. F.3.

One of the benefits of this approach is the insight gained by the ability to visualize any architecture in the feasible set. By assessing these architectures for various parameters, such as complexity, a great deal of learning about the design process can be gained. For example, Fig. F.4 depicts the set of feasible architectures that were indentified in a study of fighter aircraft subsystems. Over 20,000 possible architectures were identified that could be constructed to create the aircraft under study. When these architectures were then assessed for complexity using a variant of Eq. F.1, and subsequently ranked from lowest to highest complexity, the results were as shown in Fig. F.4. Superimposed on this graph are the complexity numbers calculated for both the F-18 and F-22 fighter aircraft. It is interesting to note that the F-18, which has demonstrated a very stable cost basis during its lifetime, lies in an area of the graph that exhibits a low slope of the cost/complexity trend. This indicates that small changes in complexity have very little impact on system cost. In contrast, the F-22 aircraft is not only very high on the complexity curve, but is also in an area with a very high slope. This indicates that even small increases in complexity lead to dramatic increases in cost.

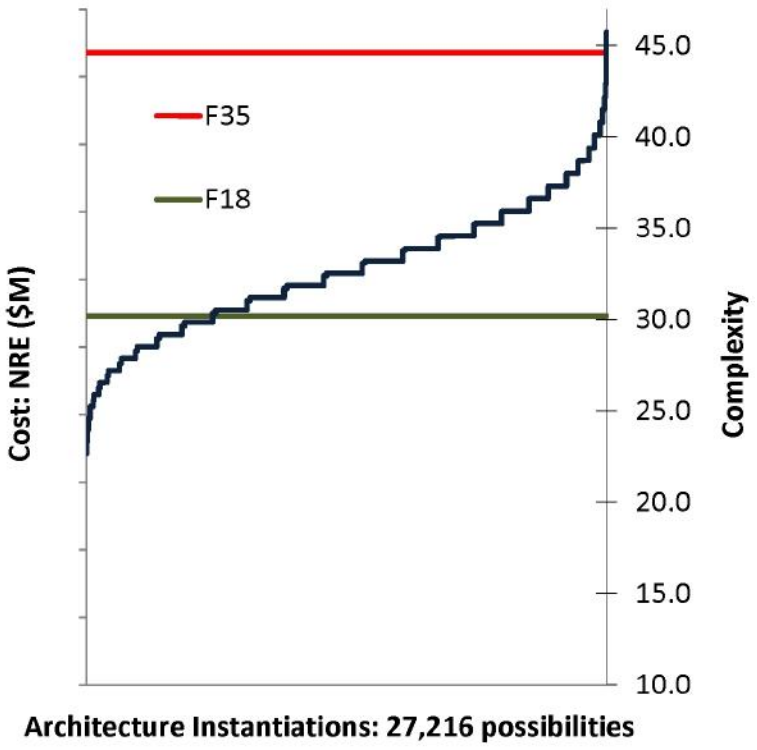


Figure F.4: Complexity and cost for various architectural possibilities

Further detail on the filter-based method of architectural enumeration is presented in section F.5.

Assessing Architectural Uncertainty

It is generally believed that there is a monotonic relationship of increasing cost to increasing complexity. Less clear is the relationship of uncertainty to complexity. Is it possible to architect a system that is highly complex and yet highly certain in its behavior, and vice versa? This question is important as low TRL technologies are planned for insertion into development programs that may span a decade or more. It is also important as an additional parameter to consider in the design process when multiple architectures may exhibit similar levels of complexity. Figure F.5 presents the feasibility (defined here as the likelihood of meeting design requirements) of various architectures studied in this work. Of note is the fact that all-electric architectures have a significantly higher level of likelihood that they will be unable to meet all system requirements (thermal requirements in this case). There are a number of factors that contribute to this assessment, and they are presented in section F.4. Robustness, another system characteristic closely related to uncertainty, is also critical to the long term sucess of a vehicle platform. Robustness can be viewed as the ability of a system to continue to meet its design intent as it experiences both internal

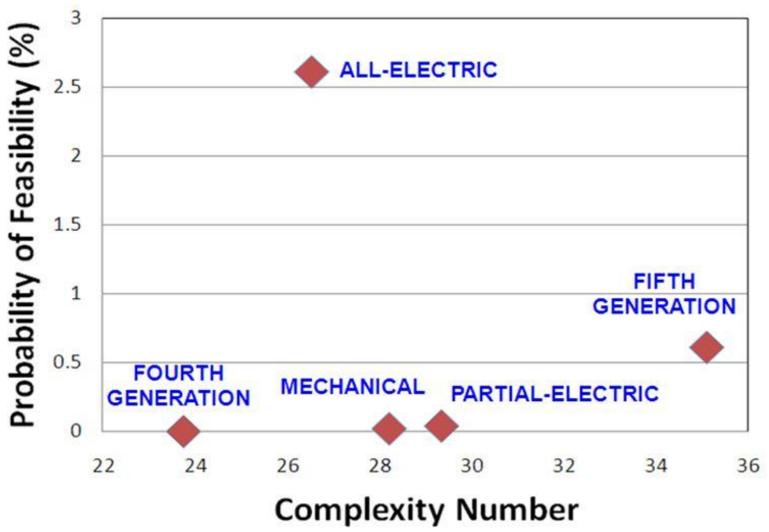


Figure F.5: Feasibility and complexity for 4th and 5th generation fighter aircraft

and external perturbations. Section F.4 describes uncertainty and feasibility of architectures and provides detail on the assessment of these characterisitics.

F.2 Design System for Managing Complexity in Aerospace Systems

Design System for Managing Complexity in Aerospace Systems

Sandor Becz[1], Alessandro Pinto[2], Lawrence E. Zeidner[1], Ritesh Khire[3], Andrzej Banaszuk[4], Hayden M. Reeve[5]
United Technologies Research Center, East Hartford, CT, 06108

As defense systems become increasingly complex, cost and schedule overruns become increasingly problematic. With reduced budgets, smaller acquisitions, and multiple competing programs, the Department of Defense struggles to meet the needs of current and future force requirements. At the same time, systems engineering practices have not kept pace with the increase in performance requirements typical of today's defense platforms. In order to better manage the exploding complexity of these systems, a new design paradigm is clearly needed. The use of abstraction and model-based methods can provide the formalism required to meet these needs. This paper presents an overview of a notional design system to provide these tools and discussing the benefits of this approach.

I. Introduction

The past several decades have seen the introduction of significant technological and architectural changes in aerospace systems in an effort to improve the performance and capability of new platforms. For example, the 787 and F-35 have incorporated more-electric systems for functions such as cabin pressurization and flight control actuation. The push to lower maintenance cost and increased dispatch reliability has lead to the adoption of prognostics and health management (PHM) systems. The implementation of fly-by-wire for flight control has reduced weight and pilot work load on many platforms. However, these and other driving forces have lead to an exponential growth in the complexity of modern aerospace platforms and accompanying design and development challenges.

The fifth generation F-35 tactical fighter offers a good example of the increased capability these new systems provide and also the resulting challenges. The F-35 is the only fighter capable of transitioning from vertical flight to supersonic cruise. It offers superior survivability through the incorporation of features such as internal stores and a composite airframe. In short, it offers 3-8 times the operational capability of fourth generation aircraft such as the F-16 or F-18 while providing superior range. This capability, however, has come at the cost of increased technical complexity. For example, the F-35 has 130 subsystems, order 10^5 interfaces, and 90 percent of its functions managed by software[1]. This is a substantial growth from the F-16 that has 15 subsystems, order 10^3 interfaces, and less than 40 percent of its functions managed by software[1]. Greater electrical loads from avionics, power electronics, and airframe actuation have increased power management requirements and more closely coupled systems together. These increased avionics and electrical loads, when combined with increased engine and actuation heat loads have increased the demands on the aircraft thermal management system while the thermal constraints of a composite airframe and limited ram cooling make it harder to get heat off the aircraft.

These increases in technical complexity have been accompanied by increases in complexity of system requirements and organizational partnerships. Complexity in requirements stems from the need to meet multiple present and future mission objectives. The F-35 design meets multi-role objectives through three different variants: Short Take Off and Vertical Landing (STOVL), Carrier Variant (CV), and Conventional Take Off and Landing (CTOL) operations (Figure 1). This requires that the center body of the F-35 be designed to meet a wide range of requirements. There is also increasing complexity in the development team. The F-35 is being funded by nine partner nations and being developed by a broad multinational team. Increasing complexity in requirements and

[1] Staff Engineer, Thermal Management, AIAA Senior Member.
[2] Senior Engineer, Embedded Systems & Networks, AIAA Member.
[3] Senior Engineer, System Design & Integration, AIAA Member.
[4] Fellow, Systems Department, AIAA Member.
[5] Staff Engineer, Thermal Management, AIAA Member Senior.

development teams is not solely a challenge for government funded development programs. The 787 is a good example of a multi-national development team and supply chain and the challenges associated with this.

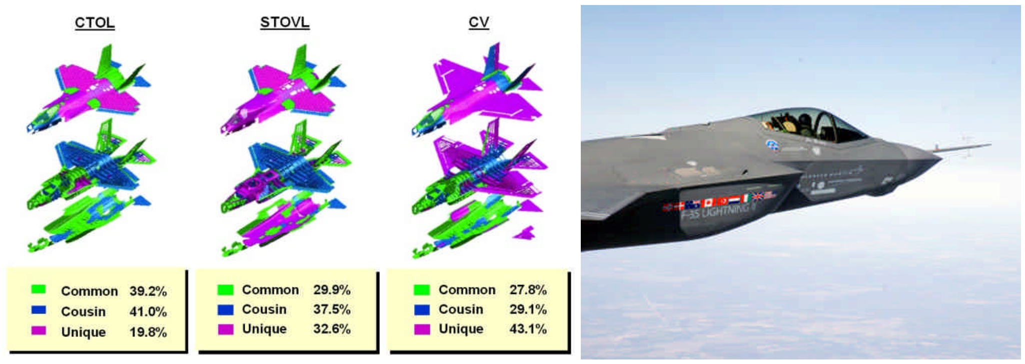


Figure 1: Illustration of the three F-35 variants (left) and F-35 in flight

These tightly coupled aircraft subsystems often have very different length and time scales and the dynamic nature of their interactions, coupled with their inherent uncertainty, drive system behavior that is often difficult to predict during conceptual and preliminary design stages. This lack of understanding results in so called “emergent behavior” or functionality that is not intended but realized during validation and verification of the system. For example, the F-35 has required redesign of its 270 VDC electrical system following issues identified during flight testing[2] and its thermal management system, which meets requirements, may need to be redesigned to provide more thermal margin[3]. The A400M provides further example of complexity issues, in this case stemming from development and integration of the Full Authority Digital Engine Control (FADEC)[4]. The complexity of new aerospace systems have contributed to unexpected development costs and delays on both military programs (F-35 delayed 1-3 years, A400M delayed 3 years) and commercial programs (787 delayed 28 months, A380 delayed 18 months).

Analysis conducted by the US Government Accountability Office (GAO) of major defense acquisition programs found that research and development costs are on average 42% higher than originally estimated and that the average delay in delivering initial capability to the war-fighter is 22 months[5]. Analysis by the RAND Corporation found that the largest component in the growth in the cost of fixed wing aircraft has come from increased complexity[1]. As systems have become more complex they have become not only more expensive to develop, but the ability to predict that development cost is inaccurate. Managing and minimizing the complexity of new system development offers the ability to reduce both the magnitude and unpredictability of development cost. The GAO found that development programs that had more knowledge earlier in the development cycle incurred reduced cost overruns. Specifically: programs that start development with fully mature critical technologies experienced 30% less R&D cost growth; programs that held system engineering reviews (requirements review, functional review, or preliminary design review) prior to development start experienced 20% less cost growth; finally, programs that had no changes in key performance parameter requirements had one third the cost growth of other development programs.

To meet the challenge of developing future complex systems in a cost-effective and predictable manner a new generation of design processes and tools are required that manages complexity at multiple levels. In many aerospace subsystems the design architecture and technologies have not changed for decades. The arrival of new architectures and technologies has not been accompanied by a commensurate advancement and adoption of the design processes and tools needed to develop these very complex systems. The next section presents a new design paradigm and discusses key needs and potential tools that must be created to address these needs. The unifying theme of these tools is to attain knowledge earlier in the design stage in order to better identify the scale, impact, and behavior of system interactions early in the design process in order to ensure complexity is understood before key design decisions and development investments are made.

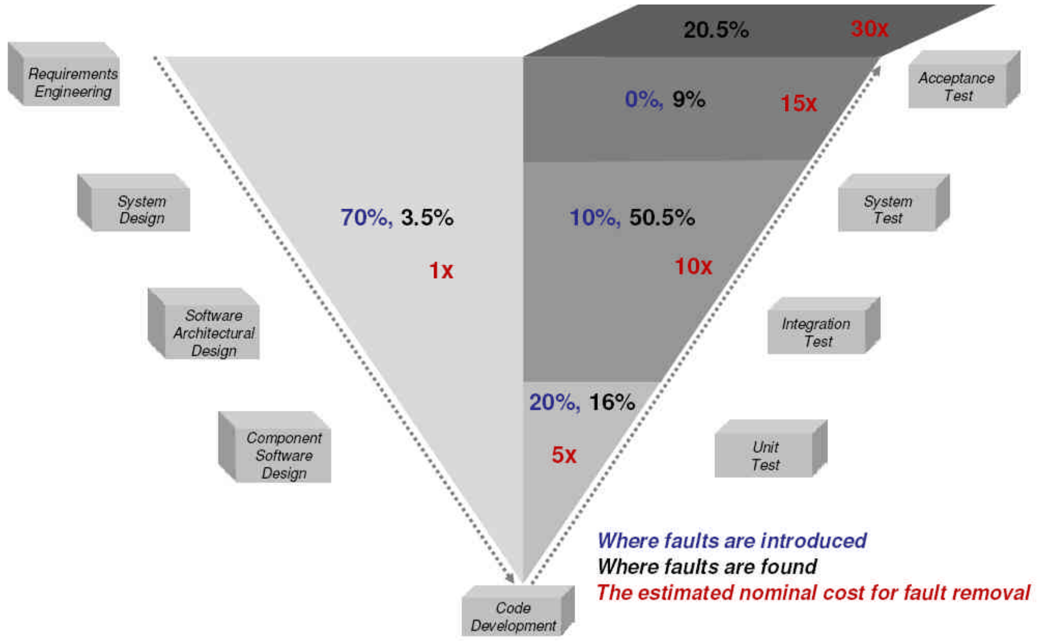


Figure 2: The traditional design 'V' showing an estimation of the introduction, detection, and cost of removal of faults during software development

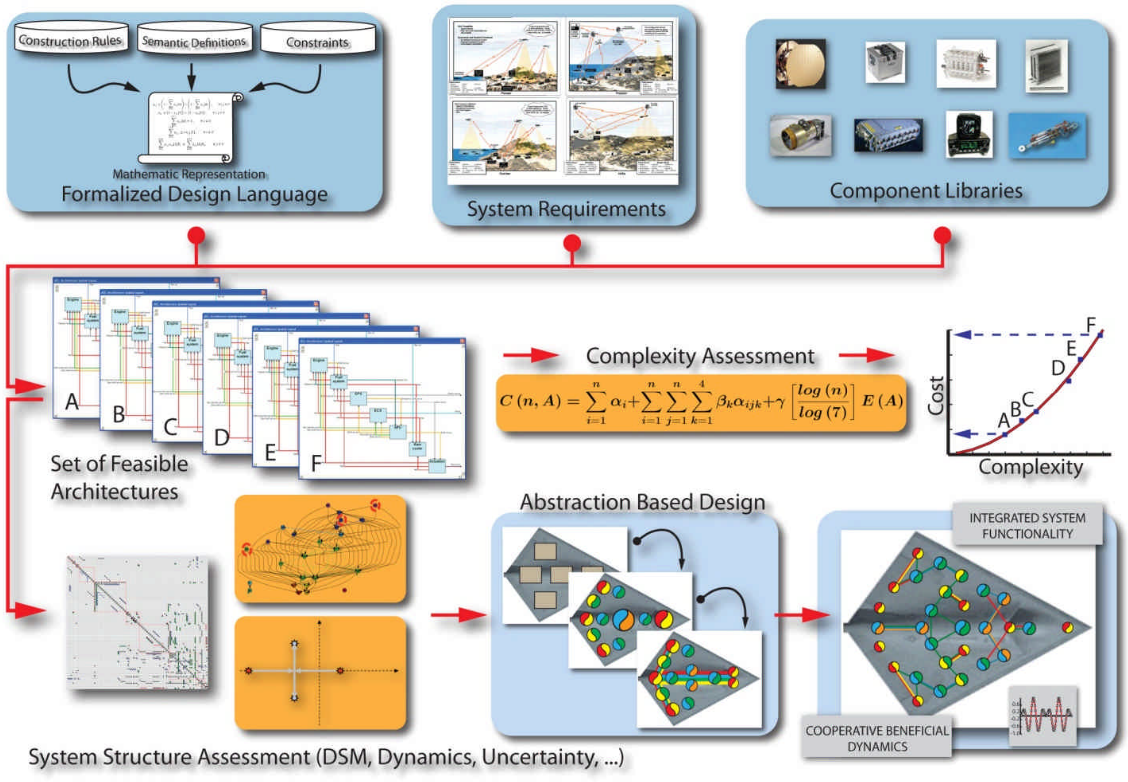

Construction Rules
Semantic Definitions
Constraints
Mathematic Representation
Formalized Design Language
System Requirements
Component Libraries
A
B
C
D
E
F
Set of Feasible Architectures
Complexity Assessment
$C(n,A)=\sum_{i=1}^{n}\alpha_i+\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{4}\beta_k\alpha_{ijk}+\gamma\left[\frac{log(n)}{log(7)}\right]E(A)$
Cost
Complexity
Abstraction Based Design
INTEGRATED SYSTEM FUNCTIONALITY
COOPERATIVE BENEFICIAL DYNAMICS
System Structure Assessment (DSM, Dynamics, Uncertainty, ...)

Figure 3: Proposed design process for complex systems

II. Overview of Design Process for Complex Systems

The design approach employed within the defense industry today is one of sequential flows of information separated along functional lines. For example, power system design activity flows from top level requirements which exist separately from hydraulic system requirements, and so on with each major subsystem. Within each system domain, the architectural layout of components is typically performed first with the overall functionality and performance determined through analysis of the selected architecture. Once the primary components have been assembled the supporting systems such as controls and communication are then developed around this layout. This approach of subdivision and refinement into more detailed design representations results in a 'design V'. This paradigm has been applied widely in many industries to organize the development process and in many applications the sequential development and refinement from concept to preliminary design to detailed design is functional. For complex systems with a great deal of interaction this compartmentalized design process leads to significant amounts of rework late in the design phase due to issues related to performance shortfalls as well as unpredicted emergent behavior resulting from various system interactions. The cost of the late detection of faults during software development has been characterized by Bruce Lewis of Army ARMDEC using data from a NIST analysis of the economic impact of inadequate infrastructure for software testing[6] (Figure 2). The ability of subsystem suppliers to pass requirements and static boundary conditions "over the wall" to other subsystem suppliers is no longer acceptable. Decisions made in isolation respective to one system (power distribution, for example) have repercussions on all other systems due to emergent behavior.

Given these limitations inherent in today's design processes, a new design approach is required that incorporates not only individual subsystem functional performance, but also all of the dynamic interactions between these systems. In addition, control and communication must be considered early in conceptual design as primary aspects of equal importance to meeting platform performance. To maintain tractability, abstraction must be introduced into the design flow to expose only those elements relevant to various stages of design, but without sacrificing the link between design choices and system performance requirements. Synthesis techniques that automatically explore the design space in search of architectures with improved performance must also be included due to the limitations of humans to manually search the large number of configurations possible.

A candidate design system is shown in Figure 3. System requirements and component libraries (representing the existing knowledge of constitutive parts of potential solutions) are formally represented in a formal design language such as AADL, UML, and SysML [reference]. This information feeds a design process that has four key elements:

1. **Abstraction Based Design Tools:** Provides a design and evaluation framework that can model complex heterogeneous systems. Enables abstraction of system models to allow complex interactions (e.g., controls and communications) to be assessed at high level of abstraction early in the design cycle. Incorporates verification and 'correct by construction' elements.
2. **Quantitative Complexity Metrics:** Quantification of complexity to provide a metric that can guide early design decisions and identify sources of complexity within candidate architectures. Complexity quantification needs to be abstracted to enable its use from early conceptual design through product development.
3. **Advanced Architecture Synthesis Methods:** An advanced set of tools that enables the formal and automated architecture synthesis, enumeration, and evaluation of all feasible architecture options, and decomposition and clustering of architectural elements in order to minimize complexity propagation in the system.
4. **Robust Uncertainty Management:** Advanced tools to access the interplay between complexity and uncertainty. Enabler in identifying strong coupling between systems with high uncertainty, thereby allowing the identification and management of key risks.

The following sections address each area in more detail, discussing the key requirements of these approaches and novel elements.

A. Abstraction Based Design

To meet the needs of future platform developers, abstraction of the design space is necessary to allow early conceptual efforts to progress quickly without sacrificing the ability to determine if performance requirements are satisfied. For example, as architectures become increasing distributed and heterogeneous, reliability and fault tolerance become primary design factors that can no longer be relegated to a "post-assessment" once systems design is complete. Therefore, a means of quickly decomposing systems and analyzing their interconnections and the effect of cascading failures must be developed to facilitate the exploration of extremely large design spaces.

The methodology known as Platform Based Design (PBD)[7] is a key intellectual framework to utilize in product development in situations where there is a great heterogeneity of subsystems – different physical, computational and communication subsystems and choices for architectural implementation. PBD also addresses situations where there is large scale both in state and physical distribution as well as the situation where the subsystems are integrated in that there is significant interaction that must be recognized and exploited in order to meet performance and cost targets.

There are two key principles behind PBD to address issues in the development of complex systems for defense applications. PBD addresses complexity by introducing into the design process layers of abstraction that provide appropriate fidelity for effective design decision making while bringing forward implementation constraints into the design early in the overall process thus enabling early verification. PBD also separates out the specification of the functionality and the architecture and at each layer of abstraction maps the required functionality to a chosen architecture and then refines the choices at each layer. This separation enables the effective trade study and capture of requirements versus cost and performance and the reduction of overall complexity.

PBD methodology was successfully adopted by several automotive manufacturers in Europe for managing development time and cost as this design process enables management of increased complexity, enables software reuse, and reduces the verification and validation effort. However, there is significant investment needed to extend Platform Based Design to design of military aerospace systems because of the additional system complexity, performance and safety requirements well beyond and above those of the commercial car industry.

Effective verification of system level performance is a key to delivering high performance and cost effective solutions to the defense contractor base. PBD is a framework for the effective deployment of tools for this verification to be done either through a “correct by construction” approach or through the deployment of verification tools throughout the design process. The key enabling technology is twofold: first to insert appropriate verification in the abstraction layers early in the design process and second to deploy robust design tools to quantify and mitigate the effects of uncertainty.

The creation of abstraction layers as defined in the PBD methodology is a key to controlling complexity and producing a truly scalable design methodology. At each layer of the process design space exploration is used to create a rich set of alternative architectures to meet required functionality, moreover, it is critical to verify the design with constraints that will be seen at lower levels so that “large loop” re-designs are avoided. PBD thus offers the framework for effective design. An example of how PBD might be applied to the architecture synthesis of an aerospace electrical system is provided in Reference [8].

B. Characterization and Quantification of Complexity

A key element to the proposed design system is the ability to quantify the complexity of advanced systems throughout the design process. This requires both abstract (i.e. low fidelity) complexity metrics to serve as a leading indicator of complexity for use early in the process during configuration selection, cost estimation, and bidding and proposing, all the way to detailed complexity metrics for use during detailed design. There has been considerable work defining complexity but less effort on constructing a quantitative metric that can guide design decisions. Kim and Wilecon[9] provide a review of complexity definitions that have been developed. These definitions cover the range of project, product, R&D/innovation, integration, and market areas and the definitions include the following core elements:

- *Numbers:* Number of different disciplines or departments involved. Number of parts, technologies, or functions required in a product.
- *Degree of Interdependency:* Level of interdependency among the domains, functions, or disciplines involved.
- *Intricacy or difficulty:* Novelty of project (minor modifications and growth and derivative versions versus clean sheet designs with untested technologies)
- *Limitations:* A compounding factor that can increase the complexity in the areas above. Examples include: limited time to market, tight performance requirements (weight, thrust), stringent constraints (thermodynamic limitations).

Other definitions have been used in the aerospace field. AFRL’s INVENT program[10] views complexity as equivalent to the inflexibility of a design to meet future growth requirements. That is, how tightly integrated the design space is with respect to change. Arena et al.[1] used the term complexity loosely to refer to the increased capability of aircraft. Jones et al.[11] developed a quantitative metric to estimate the cost of large scale systems by developing a metric based on the number of nodes and links within a system.

To enable a design process that utilizes complexity as a primary driver, an appropriate definition of complexity is clearly needed. There are many options for defining this quantity, some more useful than others. To be useful, the resultant equation should consist of parameters that are readily available to the designer. While many previous works have considered part count and/or interconnects as an indicator of complexity, this significantly oversimplifies the problem. It is of course possible to design a system that consists of many interconnects, and yet is not complex. Conversely, a system that performs highly complex functions, perhaps with the aid of software, may have relatively few components or interconnects. Also of importance is the nature of the substance that is passed from component to component, particularly in an air vehicle. For example, hydraulic interconnects are typically more complex that pneumatic interconnects due to the nature of the respective fluids and the pressures at which they operate.

In addition to physical interconnects and components, the functional behavior and interaction between components and subsystems is another element of complexity. The potential for dynamic instabilities increases as systems become more interconnected and time scales become increasingly separated. While a single, robust method for capturing these behaviors is still under development, a general framework is defined here.

$$C(n, A) = \sum_{i=1}^{n} \alpha_i + \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{4} \beta_k \alpha_{ijk} + \gamma \left[\frac{\log n}{\log 7} \right] E(A) \tag{1}$$

The overall representation for complexity developed under this effort is shown in Eq. 1. The first term represents the number of components in the system under consideration. Note that this is not simply a summation, but a weighted sum that captures inherent complexity associated with an individual component. This is an important element to capture to enable tradeoffs where component reduction is achieved at the expense of fewer, more complex components. An example of this can be seen in the trend toward reduced airfoil count in gas turbines. Turbine designers have been moving toward reduced airfoil counts for years in attempts to reduce cost. However, reducing the number of airfoils in a turbine requires that each airfoil now produce more thrust. In addition, each blade will now be heavier with associated increases in hub stress. Therefore, the design of each airfoil has become more complex in order to meet the desired goal of reduced blade count. This inherent complexity is captured by the α_i terms in Eq. 1.

The second term in Eq. 1 represents the number of interconnects in the system under consideration. Again, however, this is not a simple summation. Also reflected is the inherent complexity of each interconnection. As mentioned previously, passing various quantities between parts will have different levels of complexity associated with them. Other elements of each interconnection can be captured in this formulation as well, such as overall length. These properties are important to capture to reflect manufacturing related elements that are important to overall system complexity.

The final term in Eq. 1consists of two quantities: one that represents the level of abstraction currently under consideration ("n"), and the other termed "graph energy". The reader will note that the log terms in this representation include a log 7 in the denominator. This value is set based on the assumption that from the highest to lowest level of abstraction there are seven layers. This is based on a heuristic decomposition of typical aerospace vehicles, and may need refinement for other systems under consideration.

The graph energy term E(A) captures a number of potentially important elements of a system. At a high level, this term represents the density of connections in the system with respect to the total number of possible interconnections. The measure of this quantity can be determined in a number of ways. The one we choose here is to take the summation of the singular values calculated from the adjacency matrix of the system:

$$A = U \sum V^T = \sum_{i}^{N} \sigma_i (u_i \cdot v_i) \tag{2}$$

To effectively manage complexity in the future, domain specific standard measures of complexity are needed that would allow competing offerings to be ranked, similar to the capability-to-cost index (CCI) used to quantify the goals for future gas turbine engine performance. More importantly, identifying the key attributes and contributing factors that create or amplify complexity (and therefore development cost and risk) is a key requirement to being able to manage and minimize complexity. United Technologies Research Center (UTRC) has started to explore different methods and constructs to quantify complexity for aerospace systems[12]. This work investigates candidate system attributes that could be used quantify complexity.

C. Advanced Architecture Synthesis Methods

Advanced architecture synthesis methods are required to meet three emerging challenges. The first challenge stems from next generation systems becoming more and more complex and multi-disciplinary, resulting in the design process no longer being able to rely on the intuitive expertise of a small number of designers to make the initial down-selections to create a design space that is tractable for current methods. A formal process is required to synthesize architectures to ensure all known requirements and interactions are acceptable. Furthermore, the rate of technological advancement and complexity of these systems has increased the design configuration trade space. UTRC has developed the Architectural Enumeration and Evaluation (AEE) framework to enable the efficient and traceable decision making that rapidly reduces the entire design space to regions that warrant further investigation[13]. AEE provides a rigorous, efficient, and exhaustive means to explore a very large number of technology and architecture configurations for new application areas. AEE can tackle trade spaces that have millions or billions of design configuration possibilities and efficiently consider all possibilities to identify the set of feasible configurations (typically numbering in the thousands) and the set of promising concepts that are worthy of higher fidelity investigation (typically 10-100).

The second challenge involves superior evaluation of architecture options early in the design cycle. This will be achieved by the evaluation of both PBD domain models and a complexity metric (at appropriate levels of abstraction) during architecture evaluation and selection.

The third challenge is to understand how the architectures can be partitioned into sub-domains (so organizational entities can design and develop them) in a way that minimizes the propagation of complexity between the sub-domains and therefore minimizes likely development risk and cost. The complexity of each architecture depends not only upon its constituent technologies and its interconnections, but also on how and to what degree the architecture is organized hierarchically into modules. Each architecture can be decomposed into many different hierarchical configurations of modules, each with its own degree of complexity. UTRC has explored a method that uses a spectral graph partitioning algorithm recursively to determine the hierarchy of modules based on the limited information available at this early stage in the design process[14].

D. Methods to assess the impact of uncertainty throughout the design process

Introduction of immature or new technologies introduces significant uncertainties during the development of complex systems. This uncertainty is manifested as the lack of accurate characterization of the subsystems during early design and selection phase. This is in addition to other well known sources of uncertainties such as environmental conditions and evolving system requirements. With the variability and uncertainty associated with parameters in complex systems, a formal treatment of their impact on emergent behavior must be included in any new design paradigm. This component is virtually non-existent in current design systems, especially in the early phases of design.

It is generally believed that there is a monotonic relationship of increasing cost to increasing complexity. Less clear is the relationship of uncertainty to complexity. Is it possible to architect a system that is highly complex and yet highly certain in its behavior, and vice versa? This question is important as low TRL technologies are planned for insertion into development programs that may span a decade or more. It is also important as an additional parameter to consider in the design process when multiple architectures may exhibit similar levels of complexity. Figure presents the feasibility (defined here as the likelihood of meeting design requirements) of various architectures studied in this work. Of note is the fact that all-electric architectures have a significantly higher level of likelihood that they will be unable to meet all system requirements (thermal requirements in this case). There are a number of factors that contribute to this assessment, and they are presented in Khire, et al[15].

UTRC has assessed the impact of uncertainties on complex system selection[15]. This work demonstrated, through numerical simulation, that selection of good system architecture is critical to minimize the vulnerability of complex system to above mentioned uncertainties. In other words, the selection of apt complex system architecture will allow it to fulfill all the functional requirements. At the same time, the system will be robust against uncertainties, potentially resulting in minimum development time and resource investment.

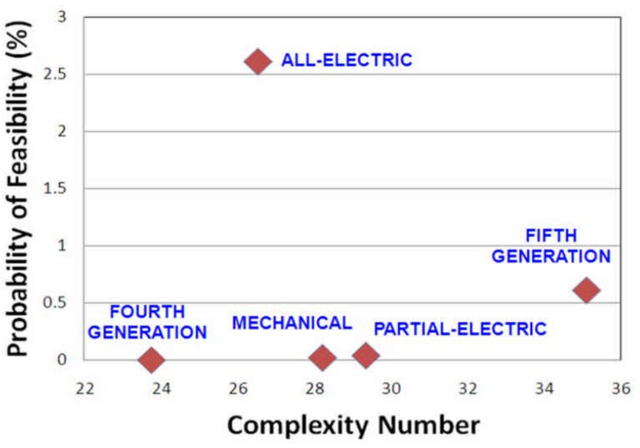


Figure 4: Complexity and uncertainty may have counterintuitive relationships.

E. Complexity as a Design Metric

One of the benefits of a complexity-based design goal is the insight gained by assessing the pattern of complexity growth within the design space. Utilizing the architecture enumeration methods mentioned in Section C, a set of over 20,000 possible architectures were identified that could be constructed to create an advanced fighter aircraft. When these architectures were then assessed for complexity using a variant of Eq. , and subsequently ranked from lowest to highest complexity, the results were as shown in Fig. 5. Superimposed on this graph are the complexity numbers calculated for both the F-18 and F-22 fighter aircraft. It is interesting to note that the F-18, which has demonstrated a very stable cost basis during its lifetime, lies in an area of the graph that exhibits a low slope of the cost/complexity trend. This indicates that small changes in complexity have very little impact on system cost. In contrast, the F-22 aircraft is not only very high on the complexity curve, but is also in an area with a very high slope. This indicates that even small increases in complexity lead to dramatic increases in cost.

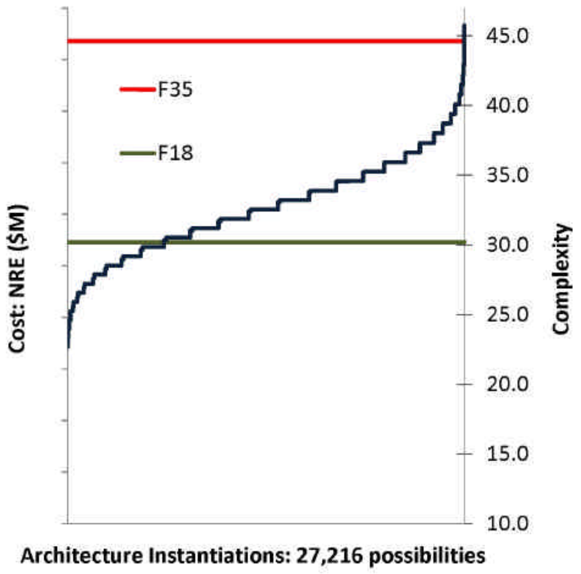


Figure 5: Cost and complexity numbers for a notional fighter aircraft study conducted using architectural enumeration methods.

III. Concluding Remarks and Future Work

A. Concluding Remarks

This paper has presented a notional design system with the intent of managing the exponentially growing complexity of heterogeneous defense platforms. As subsystems become more tightly coupled, performance requirements continue to increase, and software becomes an ever greater enabler of system functionality, the development of new methods capable of dealing with the challenges associated with designing these systems is critical. Platform Based Design provides an intellectual framework on which to construct such a system. By formally utilizing abstraction in the design space, multiple levels of fidelity can be leveraged to optimize system functionality. In addition, model-based methods coupled with architecture synthesis and analysis techniques allow a much richer design space to be explored, leading to more optimal and robust designs.

B. Future Work

To realize the design paradigm depicted in Fig. 3, several developments are necessary. Both Platform Based Design and model based design methods must be developed that allow for heterogeneous cyber-physical systems to be described formally. The semantics required to construct a language that allows these domains to be combined in a single design environment must be created to allow these model-based methods to operate. In addition, an overarching framework that ties the tools that create these models together must be developed. This capability could potentially lead to a consortium-based approach to the design of future defense systems that allows for model exchange and offers to significantly reduce the cost and schedule required to design highly complex cyber-physical systems.

Acknowledgments

The authors would like to thank Mr. Paul Eremenko of DARPA's TTO office for funding this research. This work was performed under contract FA9550-07-C-0024.

References

[1]Arena, M. V., Younossi, O., Brancato, K., Blickstein, I., Grammich, C. A., "Why Has the Cost of Fixed-Wing Aircraft Risen?" RAND Corporation, 2008, http://www.rc.rand.org/pubs/monographs/2008/RAND_MG696.pdf

[2]Boeder, J., "F-35 JSF Hit by Serious Design Problems", Daily Industry Daily, December 2007, http://www.defenseindustrydaily.com/f-35-jsf-hit-by-serious-design-problems-04311/ [cited 27 January 2010]

[3]Perrett, B., "F-35 May Need Thermal Management Changes", Aviation Week, March 12 2009, http://www.aviationweek.com/aw/generic/story.jsp?id=news/F35-031209.xml&headline=F-35%20May%20Need%20Thermal%20Management%20Changes&channel=defense [cited 27 January 2010]

[4]http://www.independent.co.uk/news/business/analysis-and-features/the-euro20bn-plane-that-may-not-fly-1866040.html

[5]http://www.gao.gov/new.items/d09326sp.pdf

[6]NIST Planning Report 02-3, "The Economic Impacts of Inadequate Infrastructure for Software Testing", May 2002. http://www.nist.gov/director/prog-ofc/report02-3.pdf

[7]Sangiovanni-Vincentelli, A., "Quo Vadis SLD: Reasoning about Trends and Challenges of System-Level Design". *Proceedings of the IEEE*, Vol. 95, No. 3, pp. 467-506, 2007.

[8]Pinto, A., Becz, S., Reeve, H. M.,"Correct-by-Construction Design of Aircraft Electric Power Systems", To be presented at 10th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference, Fort Worth Texas, 2010.

[9]Kim, J. and Wilemon, D. "An empirical investigation of complexity and its management in new product development", Technology Analysis & Strategic Management, Vol. 21, Issue 4, 2009, pp. 547-564.

[10]Walters, E. A., and Iden, S., "INVENT Modeling, Simulation, Analysis and Optimization", AIAA Aerospace Sciences Meeting, Orlando, FL, 2010-287.

[11]Jones, R., Hardin, P., and Irvine, A., "Simple Parametric Model for Estimating Development (RDT&E) Cost", 2009 ISPA/SCEA Joint Conference.

[12]Zeidner, L. E., Becz, S., Khire, R., Reeve, H. M., "Design Issues for a Bottom-Up Complexity Metric Applied to Hierarchical Systems ",To be presented at 10th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference, Fort Worth Texas, 2010.

[13]Zeidner, L.E., St. Rock, B.E., Desai, N.A., Reeve, H.M., and Strauss, M.P., "Application of a Technology Screening Methodology for Rotorcraft Alternative Power Systems," *AIAA Aerospace Sciences Meeting*, AIAA-2010-1505, Orlando, FL 2010.
[14]Zeidner, L. E., Becz, S., and Banaszuk, A., "System Complexity Reduction via Spectral Graph Partitioning to Identify Hierarchical Modular Clusters ",To be presented at 10th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference, Fort Worth Texas, 2010.
[15]Khire, R., Becz, S., Reeve, H. M., Zeidner, L. E., "Assessing performance uncertainty in complex hybrid systems", To be presented at 10th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference, Fort Worth Texas, 2010.

F.3 Correct-by-Construction Design

Correct-by-Construction Design of Aircraft Electric Power Systems

Alessandro Pinto *

United Technologies Research Center, Inc., Berkeley, CA

Sandor Becz † and Hayden M. Reeve ‡

United Technologies Research Center, East Hartford, CT, 06108

We formalize correct-by-construction design following the principles of Platform-Based Design.[1] The design proceeds by refinement steps. At each step, a specification is given in terms of requirements that an implementation must satisfy. The implementation choices are implicitly captured by a set of components and their composition rules. A class of candidate implementations of the specification is derived by formulating and solving an optimization problem. The implementation becomes the specification for the next step in the design flow. We present the trade-offs in the selection of the abstraction layers. We show how the methodology can be applied to the design of electric power systems by decomposing the design flow into the following steps: generator selection, generation of the connection configuration under faults, and topology design of the power distribution system.

I. Introduction

The design flow used today for electrical systems is mainly top-down and provides limited ability to predict, early in the design process, the consequences of radical departures from known designs on the overall performance. The design of aircraft secondary power systems has been a derivative process where previous designs that are known to work under go slight modifications to accommodate new features. Through the end of the Second World War 28 VDC systems were typical. As loads increased with the advent of the jet age the more weight efficient 115 VAC / 400 Hz distribution system became the standard.[2] For the next four decades this system dominated, typically using constant speed devices (CSDs) to ensure the constant 400 Hz frequency and 2 or 4 channels. Given the prevalence of this electrical power system research effort was directed mainly on the improvement of component level performance (weight and efficiency) rather than design methodologies and tools for automatic design exploration and verification. The arrival of new *"more-electric"* technologies such as electric main engine start, electrical cabin air pressurization, and electric primary flight control actuation has again increased the power demands on the electrical system and resulted in the adoption of higher voltage systems (270 VDC, 230 VAC Variable Frequency) in order to reduce distribution (feeder) weight. These changes have also brought system synthesis, evaluation, and verification challenges that are not well meet by the legacy design system. For example, the 787 has a fourfold increase in electrical power capability over the 777, threefold increase in the number of electrical buses, and a XXfold increase in the number of distribution states. Because the requirements imposed by these new applications are drastically different from the ones imposed on the previous generation of aircraft, re-use of known solutions and methods becomes inadequate. A major architectural redesign of the electric systems poses challenges to engineers that find themselves engaged in manual exploration of a large design space constrain by many, and informally captured constraints.

Typically, a new design is prototyped and tested. If the application requirements are not met, then the system is re-designed. The re-design cycle goes through the manual process of changing design decisions

*Senior Engineer, Embedded Systems and Networks

†Staff Engineer, Thermal Management, AIAA Member.

‡Staff Engineer, Thermal Management, Senior AIAA Member

and producing a new prototype (or a change in the current prototype). Re-design is not unusual and is the direct consequence of difficulties to evaluate design solutions and predict the impact of design decisions made in the early stages of the design process on the performance of the final implementation. This problem can be attributed to several reasons, among which we mention the semantic gap between the specification of the system requirements and the details of the implementation platform, and the lack of methods, tools and formal models helping designers in marching from the system requirements to the detailed system implementation.

These two factors are not independent. System requirements are captured using documentations (assisted by requirement management tools such as DOORS[3]), and manually refined into several linked documents that capture not only the partitioning of the system into sub-systems (already implying the system architecture), but also the local performance constraints that each sub-system must satisfy. However, the number of possible choices for a system architecture is large which makes this processes complex, and the solution sub-optimal at best. Perhaps, one simple complexity measure for a design can simply be defined by the number of requirements and the number of degrees of freedom in choosing the implementation. Further, because the high level architectural decision are based on non-executable (and non-analyzable) models, it is difficult to assess the behavioral properties of the system, and therefore impossible to look-ahead possible emergent behaviors arising from the composition of sub-systems. For these reasons, some tools have been developed to help engineers evaluating the fitness of an architecture to a given application. The Design Structure Matrix (DSM)[4,5] and the Architecture Design Graphs (ADG)[6] have been used in aerospace (among other fields). However, these methods provide limited capabilities for efficient design exploration at different stages of the design process.

In this article, we present a correct-by-construction methodology inspired by the Platform-Based Design (PBD)[1] methodology that has been successfully used in the automotive and consumer electronics domains . The PBD methodology provides an intellectual framework where a design flow that implements a specification proceeds through self-similar refinement steps. In this framework there is a clear distinction between the function (what the system is supposed to do, i.e. the requirements) and the architecture (how requirements are realized, i.e. the components and their interconnection that together implement the function) that allows for automatic design space exploration. Each refinement step consists in selecting a platform instance that correctly implements a specification. A platform instance is a valid composition of library elements that are characterized by their cost and performance metrics. Thus, a design step can be formalized by an optimization problem (in general multi-objective) whose solution (or set of non-dominated solutions) represents the functional specification to be implemented by the sub-sequent refinement step. This process repeats until the abstraction level is close enough to the implementation.

Key to the success of such methodology is the careful selection of the abstraction layers, i.e. the selection of the refinement steps. In fact, each step explores the design space along a subset of the axes representing the design variables. Thus, it is important to carefully prioritize the design choices and make sure that the performance and cost models are accurate enough for the level of abstraction such that design decisions can be made without compromising the quality of the final implementation. Ideally, if each refinement step is done by solving an optimization problem and if the models are accurate (with respect to the abstraction level), the verification effort can be minimal because the implementation is guaranteed to satisfy the specification by construction.

II. Preliminaries

Formal treatments of the PBD methodology have been presented using different mathematical frameworks such as agent algebra[7] and labeled graphs.[8] In this section we give a less formal description to outline the trade-offs involved in the definition of a concrete instantiation of a PBD design flow.

Consider a library of parametric components that can be instantiated and configured by selecting the values of the parameters. Each instance s of a library element (e.g. a generator or a load) has a set Q_s of associated parameters. A parameter $q \in Q_s$ denotes a metric (e.g. the rated power of a generator) that affects cost and performance of a design . Let $x_{s,q}$ be a variables associated with parameter q of component s. This variable ranges over a domain of values D_q. For instance, the availability a of a generator ranges in the closed interval $D_a = [0, 1] \subset \mathbb{R}$. A system comprises a set of component instances S and implicitly defines a set of decision variables $X = \{x_{s,q}\}_{s \in S, q \in Q_s}$ ranging over the domain $D_X = \times_{s \in S, q \in Q_s} D_q$. Parameters are very general quantities that can be used to model choices in the design of a system. For example, a binary parameter ι can be used to decide whether a component is really needed in a system or not. This parameter could be used to decided whether a system needs one or two generators. A designer may start with an instance that include two generators s_1 and s_2, and then realize that one generator is sufficient to power all loads, in which case the value of the variable $x_{s_1,\iota}$ may be set to zero to denote that generator s_1 is superfluous and can be removed from the system (we will exercise this feature later in our examples). Also, some of the variables may be assigned as a result of the specification. For example, the power required by a load is given as input to the design problem.

The design space is a subset of D_X. In fact, a platform is defined by the library and by a set of constraints called *composition rules*. For example, in some power systems, generators cannot be connected on the same bus. Therefore, the design space, i.e. the set of valid assignments of the variables X is restricted by a set of platform constraints $C_p(X)$. The functional requirements are captured by another set of constraints $C_m(X)$ that define those assignments that correctly implements the specification. For example, under all possible faults, critical loads must be always powered; the total power required by loads is provided by generators. Thus, the set of system configurations that are valid platform instances and that satisfy the specification is $C_p(X) \cap C_m(X)$. Finally, the cost of a system is in general a multi-objective function $F : D_X \to \mathbb{R}^f$. Thus, the optimal configuration problem can be written as follows:

$$\begin{aligned} \underset{X}{\text{minimize}} \quad & F(X) \\ \text{subject to} \quad & X \in C_m(X) \cap C_p(X). \end{aligned}$$

The complexity of this problem depends on the number of decision variables of the problem, i.e. $|X|$, the structure of the constraints, and the form of the cost function. If the library is defined a a very low abstraction level, with many components each characterized by many parameters, finding a solution to this problem becomes challenging. Imagine for example considering a library that includes wires, contactors, transformer-rectifier units (TRU), converters, inverters, generators, loads, batteries, circuit breakers, and all other detailed components of a typical system. The design process can be divided into refinement steps where the set X is partitioned into sub-sets $X_1, \dots, X_L$. At the i-th layer, the following problem is solved:

$$\begin{aligned} \underset{X_i,\tilde{X}_i}{\text{minimize}} \quad & F_i(X_i, \tilde{X}_i, X_1^*, \dots, X_{i-1}^*) \\ \text{subject to} \quad & (X_i, \tilde{X}_i) \in C_{m_i}(X_i, \tilde{X}_i, X_1^*, \dots, X_{i-1}^*) \cap C_{p_i}(X_i, \tilde{X}_i, X_1^*, \dots, X_{i-1}^*). \end{aligned}$$

where $\tilde{X}_i$ is a set of additional variables that are used to capture the abstraction of the variables in the sets $X_{i+1}, \dots, X_L$. These additional variables often represent virtual components. We will show an example of how the power distribution system is abstracted into point-to-point connections by introducing connectivity variables. The solution of this problem is the set of optimal values X_i^* and $\tilde{X}_i^*$. Clearly, $\cap_{i=1}^{L} C_i \subseteq C$ meaning that only feasible solutions should be explored. This formalization shows the choices that need to be made in the definition of a PBD flow, and interesting additional features that this methodology provides:

1. the set of variables X is far from being unstructured meaning that there are some additional constraints to take into account when deciding on the partition $X_1, \dots, X_L$. For example, the topology of the power distribution system results as a consequence of the decision on the number of generators and the connectivity requirements between loads and generators. By the same token, the insertion of tie and circuit breaker can only be decided after the topology of the power distribution system has been designed. Structural constraints arise naturally from the notion of refinement where sub-systems are further decomposed into sub-systems.

2. Ideally, X^* should be equal to $(X_1^*, \dots, X_L^*)$. However, this result depends on the quality of the abstraction, meaning how well the additional variables $\tilde{X}_i$, the constraints C_{m_i} and C_{p_i}, and the cost function F_i represent the lower abstraction levels. In fact, if the abstractions are not done carefully, the optimization problem solved at the i-th level may prevent the exploration of part of the design space by selecting a sub-optimal assignment of the variables in X_i.

3. Because the set of constraints C_{p_i} define the set of valid platform instances, it is possible to capture domain knowledge by restricting the class of architectures to be considered in the optimization problems. For example, it is possible to add constraints to only consider hierarchical systems divided into a primary and a secondary power distribution systems, or restrict the exploration to ring topologies only.

Moreover, if the optimal configuration of some of the components is known, those design variables can be fixed in the optimization problem and treated as constants.

4. The optimization problems could in principle be automatically derived from a model-based description of the library elements. In a virtual engineering environment, the library may also contain components that do not yet exist, allowing to play "what if" scenarios and automatically compute the requirements that such components should be able to satisfy. These requirements would be provided in the form of values for the parameters of the virtual components.

These two observations require to understand the structure of the design problem to build the right abstractions, and to use languages that allow to represent components and their refinements in a unified way. In this article we will show examples dealing with the first of these two requirements. The definition of the right language to use is out of the scope of this article but it is a well explored and evolving research field. Many system-level design languages are available that provide the required features. Among these, Metropolis,[9, 10] Rosetta,[11] Architectural Analysis and Design Language (AADL)[12] and SysML[13, 14] are all good candidates for a correct-by-construction design methodology. Contrary to other methods such as DSM,[4] we do not aim at providing a way of documenting and analyzing the interactions in complex systems, but rather providing an organized design method to overcome complexity.

III. Correct-by-construction design of Electric Power Systems

In our design problem, the specification is given in terms of a set of loads together with their power and reliability requirements. The objective is to determine the architecture of an electric power system able to satisfy the demand of the loads. We start with a qualitative analysis of the main drivers of the overall system cost with the intent to partition the design decisions and define the refinement steps.

The efficiency of a generator $\eta(P, P_l)$ is a function of the the power P offered by the generator, and the total power P_l absorbed by the loads connect to it. By fitting data from a database of representative generators, it was found that the efficiency is a concave function of P_l/P meaning that the efficiency improves when the generator is fully utilized by the loads.

Observation 1. *The maximum efficiency of a power system is achieved when the rated powers of the generators are matched to the power requirements of the loads.*

The weight of a generator is a function of the rated power. The function $w(P)$ that links the power and the weight is a concave function and can be fitted well by a quadratic function. This means that in terms of watt per pound, generators with high rated power are preferred to small generators.

Observation 2. *The minimum weight of a power system is achieved by selecting generators with as high rated power as possible.*

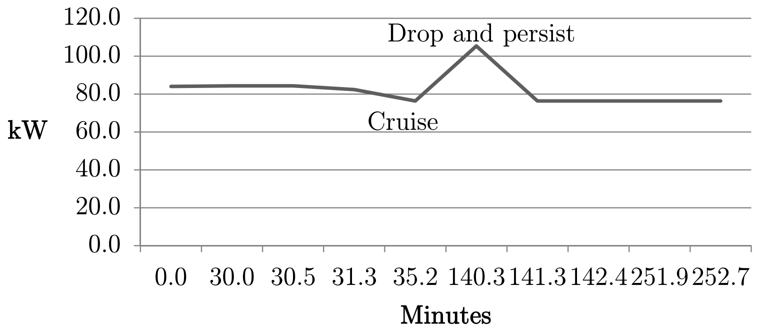


Figure 1. Power profile during a mission from take-off to landing.

To understand the trade off between efficiency and weight, consider the mission profile shown in Figure 1. In this simple UAV mission, the power consumption is not uniform over time. A peak in the power consumption, mainly due to the use of electric actuators during the persistence phase, can be observed. If we were to favour weight over efficiency, we would select a generator able to provide as much as $105kW$. However, this generator would be inefficient for the rest of the mission providing an efficiency of approximately 80%. If we were to favour efficiency over weight, then one choice would be to use two generators of 85 kW and 30 kW and use the smaller generator only in that phase of the mission where more power is required. In this case we would have a weight penalty of roughly 10 lb but without any loss in efficiency.

However, an additional metric to consider is the complexity of the power distribution system and the control and communication sub-systems required to manage redundancy and maintain the desired power quality. In fact, control complexity increases when generators are matched to the loads because of their limited authority in driving the voltage on the power buses. Further, increasing the number of generators would also required to increase the number of buses which has two effects: it makes the topology of the power distribution system more complex, and it increases the complexity of the state machines that control power transfers.

Observation 3. *The costs of the power distribution system and the control system increase for more efficient electric power systems.*

Figure III shows the qualitative trade-off in the design of the electric power system. Few large generators will provide the best solution in terms of pounds per watt and in terms of the complexity of the power distribution system, denoted by *cplx*. However, many small generators will be able to deliver a very efficient solution allowing, for example, a UAV to fly longer for the same amount of fuel, while at the same time lowering the heat rejection requirements. The number of generators affects also the overall reliability of the electric power system. The Probability Loss Of Function (PLOF) decreases with the number of generators as more sources are available to power the system loads in the event of a generator failure. In order to maintain the reliability of the system above a certain value, more components need to be added and therefore the overall cost and complexity increases.

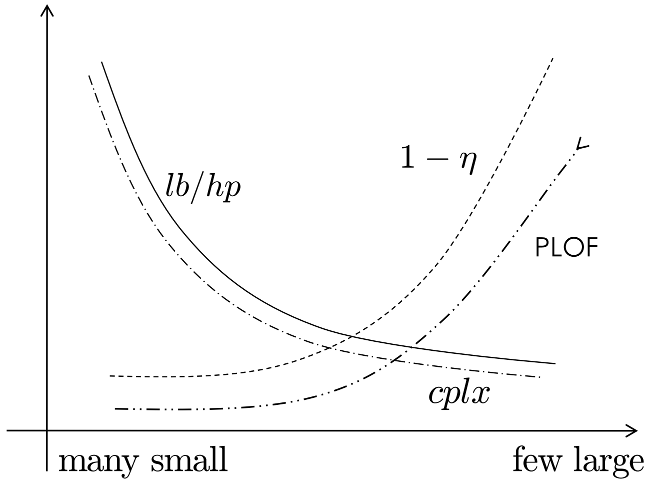


Figure 2. Trade-offs between weight, efficiency and complexity of the distribution and control systems.

From these observations, we conclude that the the selection of the number of generators and their rated powers drives the trade-off between cost and efficiency of the electric power system. It is reasonable to explore this trade-off first in the design flow. However, the cost of the power distribution system must also be taken into account. In our methodology, this objective can be achieved by including a virtual component in the library characterized by a few parameters that capture the cost and performance of the power distribution system.

Consider a power distribution sub-system that connects n generators to m loads. Because loads and generators may have different voltage interfaces, the cost of power conversion must be taken into account. This cost depends on which generator powers which load. Moreover, because of the reliability constraints imposed by the loads, each connection should provide a minimum level or reliability. The number of connections, i.e. the number of physical paths that must be provided by the topology of the power distribution system, affects the cost of the communication sub-system. Finally, the reliability levels of the loads also determine the cost of the communication sub-system as reliable connections cost more than unreliable ones. The cost model obtained from historical data shows that the weight of power conversion is a linear function of the power. Therefore, the total weight of the power conversion units is independent from the way in which loads are associated to generators. The efficiency of the power conversion units is fixed and therefore there is no trade off with weight.

Observation 4. *The cost drivers for the power distribution system are the number of generator-to-load connections and their reliability.*

Thus, the power distribution system can be abstracted by a set of parameters defining the reliability of the connections from generators to loads.

In summary, we justify the following design flow for aircraft electric power systems (depicted in Figure III):

Step 1: Generator selection . The specification is given by a representative power profile for each load together with reliability requirements. The library contains generators, loads and a virtual power distribution system. The synthesis problem is formulated as a multi-objective optimization problem that determines the size of the generators and the assignment of loads to generators such that the weight and the inefficiency of the system are minimized. Notice that the number of electrical power sources (engine driven generators, ram air turbine generator, batteries) is in general constrained by formal design rules. For example, a minimum number of power sources are required to meet safety requirements (primary flight control and cabin pressurization) and ensure high aircraft dispatch availability levels (main engine start). Furthermore the number of primary generators is almost always a multiple of the number of aircraft engines. The electrical loads are partitioned into groups based on the required power supply (28 VDC, 115 VAC, 230 VAC etc.) and the number of generation sources in use during typical operation. The power distribution system is abstracted by two set of variables: $\{y_{ij}\}$ indicating whether load i is connected to generator j, and $\{a_{ij}\}$ denoting the availability of the connections.

Step 2: Topology design . The power distribution system is refined by instantiating buses and connections among them to form an optimal topology. Variables $\{y_{ij}, a_{ij}\}$ are refined into paths in the topology. In addition to busses and contactors, power conversion devices such as transformer rectifier units (TRUs) and inverters are instantiated to ensure that the different power requirements of the loads are meet. The topology of the electrical power system distribution architecture is optimized to minimize cost (weight, inefficiency, etc) and complexity while meeting the system level reliability constraints.

Step 3: Control design . Given the topology and the paths from generators to loads, and given fault conditions of the system, a state machine can be synthesized that controls circuit breakers and tie-breakers to guarantee that critical loads are always powered.

Step 4: Embedded system design . In this last step, the control functions are implemented on a networked system that comprises a network and a set of computation resources.

The last two steps are out of the scope of this article and they will be included in our future work.

IV. Step 1 : Generator selection problem

At this abstraction level, the library provides three types of components: loads, generators and a power distribution system. Several composition rules may be associated with the platform including connection rules (generators cannot be connected to other generators, loads can be connected to generators only through the power distribution system etc.). During this design step, we enforce many of this rules by construction as it will be clear soon. However, these constraints do not disappear but are propagated down to the lower levels of abstraction (see Step 2 of the design flow in Section V).

The variables and symbols used in the definition of the optimization problem are shown in Table 1. The specification includes n loads and T mission phases. The power required by load i during phase t is denoted by $L_i(t)$. Moreover, let r_i be the reliability requirement of the i-th load. This set of variables have fixed values and capture the specification of the design problem.

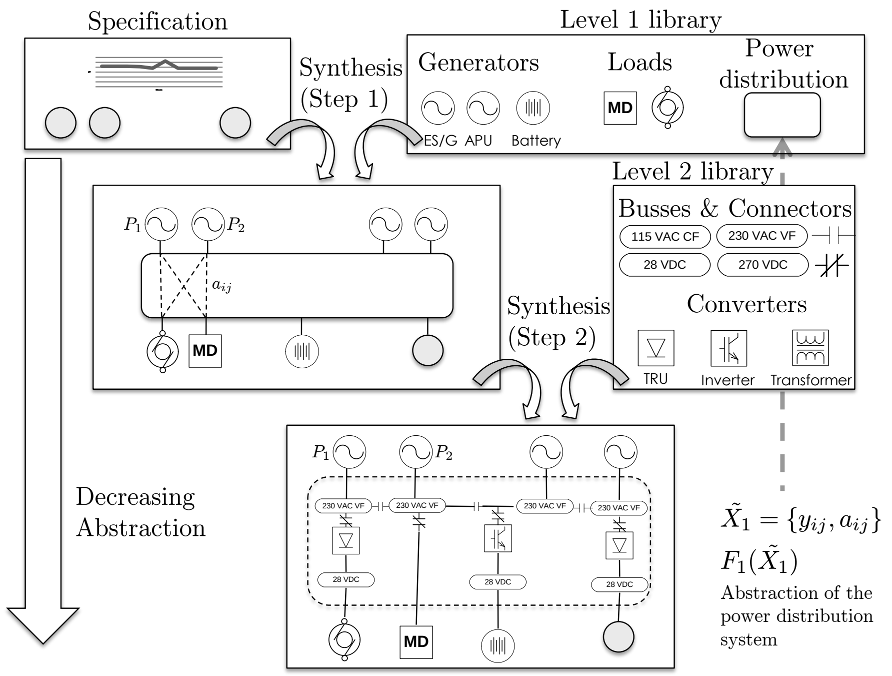


Figure 3. Detailed graphical rendition of the first two steps of the design flow.

Symbol	**Domain**	**Meaning**
i	$\{1,\ldots,m\}$	Load index
j	$\{1,\ldots,n\}$	Generator index
t	$\{1,\ldots,T\}$	Mission phase index
$L_i(t)$	$\mathbb{R}_{\geq 0}$	Power of load i at t
r_i	$[0,1] \subset \mathbb{R}$	Reliability required by load i
P_j	$[0, 330e3] \subset \mathbb{R}$	Power offered by generator j
x_j	$\{0,1\}$	Installation variable
$y_{ij}(t)$	$\{0,1\}$	Load i connected to generator j
a_j	$[0,1] \subset \mathbb{R}$	Availability of generator j
$a_{ij}(t)$	$[0,1] \subset \mathbb{R}$	Availability of connection ij at t

Table 1. Symbols used in the formulation of the optimization problem.

We start by observing that the optimization problem is formulated in terms of the least constraining platform instance, meaning a platform instance with the maximum number of generators m that a designer considers appropriate for the application. An upper bound for m is n. However, not all generators will be actually used by loads, and some of them will be removed as a result of the synthesis procedure. A binary variable x_j is used for this purpose. The value of x_j is equal to one if a generator is needed, and zero otherwise. Each generator is associated with a parameter P_j which denotes the value of its rated power. We also include the virtual power distribution system as part of the platform instance to be optimized. Binary variable $y_{ij}(t)$ is equal to one if load i is powered by generator j during phase t, while a_{ij} is the availability of the connection. The two composition rules included in C_{p_1} are the following:

$$y_{ij}(t) \leq x_j \qquad \forall\, i,\ \forall\, j,\ \forall\, t \tag{1}$$

$$a_{ij}(t) \leq y_{ij}(t) \qquad \forall\, i,\ \forall\, j,\ \forall\, t \tag{2}$$

meaning that node i can be connected to generator j only if generator j is actually present in the architecture (Constraint 1), and that the availability of a connection is zero when the connection is not active (Constraint 2)

The set of implementation constraints C_{m_1} is the following:

$$\sum_i L_i(t) y_{ij}(t) \leq P_j \qquad \forall\, j,\ \forall\, t \tag{3}$$

$$\sum_j y_{ij}(t) \geq \lambda_i(t) \qquad \forall\, i,\ \forall\, t \tag{4}$$

$$\sum_{i,j} \ln\left(1 - a_j a_{ij}(t)\right) \leq \lambda_i(t) \ln r_i \qquad \forall\, i,\ \forall\, t \tag{5}$$

where $\lambda_i(t)$ is equal to 1 if $L_i(t) > 0$ and it is equal to zero if $L_i(t) = 0$. Constraint 3 requires a generator to be able to power all loads connected to it. Constraint 4 requires that a load be connected to a generator whenever it needs power during the mission. Constraint 5 imposed that the aggregate reliability of the power sources connected to the load satisfies it reliability requirements.

The multi-objective function for this problem includes weight and inefficiency components $F_1 = (W, 1 - \eta_1(1), 1 - \ldots, 1 - \eta_m(T))$ defined as follows:

$$W = \sum_j w(P_j) x_j \tag{6}$$

$$\eta_j(t) = \eta(P_j, \sum_i y_{ij}(t) L_i(t)) x_j \tag{7}$$

In this formulation we did not consider storage elements which is part of our future work. Storage can be considered in this formulation by adding a vector of parameters $\Delta(t)$ denoting the amount of time the system spends in phase t of the mission. Energy balance constraints can the be added to the formulation. The optimization problem is mixed-integer, non-linear and multi-objective. It is therefore a hard problem to solve. In the next sections we propose some variants of the problem that can solved using standard optimization methods.

A. Problem variants

The first problem variant that we consider is to remove the dependency from variable t in the formulation of the optimization problem. Removing the time dependency has two effects. The number of decision variables is reduced by considering one configuration that satisfies either the worst case or average case scenario. The second effect is the simplification of the controllers that handle the switching of the contactors to disconnect and reconnect loads during the mission. This simplification results in a lower complexity and cost for the distribution network and software development. Together with the elimination of the variable t, it is possible to further reduce the complexity of the optimization problem by considering the reliability of connections $\{a_{ij}\}$ to be the same for all connections, say a_c. The resulting optimization problem becomes the following:

$$\begin{aligned}
\underset{\mathbf{x},Y,\mathbf{P}}{\text{minimize}} \quad & C \\
\text{subject to} \quad & \sum_i \max_i L_i(t) y_{ij} \leq P_j && \forall\ j \\
& \sum_j y_{ij} \geq 1 && \forall\ i, \\
& y_{ij} \leq x_j && \forall\ i,\ j, \\
& \sum_j y_{ij} \ln{(1 - a_j a_c)} \leq \max_i \ln r_i .
\end{aligned}$$

Perhaps, the most important abstraction that need to be sought is one that reduces the complexity of the optimization problem coming from the cost function. Consider the rated power of generators to belong to a finite set of values $D_{P_j} \in \{p_1, \ldots, p_g\}$. $\forall j$. This will allow us to define a finite set of weight coefficients $w_h = w(p_h)$ and a set of binary variables u_{jh} that is equal to 1 if generator j has rated power equal to p_h. Therefore the total weight of the architecture can be expressed as follows:

$$W = \sum_j \sum_h u_{jh} w_h \tag{8}$$

with the additional constraints that $\sum_h u_{jh} = 1$, $\forall j$., meaning that a generator can only be of one type.

This formulation does not help in simplifying the expression of the efficiency of a generator. However, a similar approach can be followed to divide the total power assigned to a generator into l consecutive intervals $\hat{L}_k = [q_k, q_{k+1}]$, $k = 1, \ldots, l$, $q_1 \geq 0$, so that efficiency numbers can be precomputed as follows:

$$\eta_{jhk} = \eta(p_h, q_k) \tag{9}$$

The inefficiency of the system is the sum $\sum_{jhk}(1 - \eta_{jhk}) z_{jhk}$ where variables z_{jhk} is equal to 1 if generator j is used (i.e. $x_j = 1$), has type h and has a total load attached to it in the interval $\hat{L}_k$. Additional constraints are required to define the variables z_{jhk}. However, this procedure can be automated and the size of each interval can be defined based on the required approximation accuracy.

With this formulation, we reduced the problem to a binary problem (i.e. one where each decision variable is binary) that can be solved using standard pseudo-Boolean solvers, genetic or evolutionary algorithms.

V. Step 2: Power distribution design problem

The input to the power distribution design problem is the set of parameter values $\{y^*_{ij}\}$ and $\{a^*_{ij}\}$, together with the specification used as input to the generator selection problem. Topology design is a known problem and can be formulated as a multi-commodity flow problem. However, we will see that a pre-processing step is needed to guarantee that the controller design problem (Step 3 not explored in this paper) is feasible[a].

Consider a set of nodes $V = G \cup L \cup B$ in the architecture of the electric power system that comprises a set G of $m^* \leq m$ generators from Step 1, a set L of n loads, and a set B of b buses, where b is an upper bound on the number of buses in the system. Further, the set of loads G is partitioned in the set of AC loads L_{AC} and DC loads L_{DC}. Similarly, the set of buses is partitioned in the set of AC buses B_{AC} and the set of DC buses B_{DC}. For $u, v \in V$, let the binary variable e_{uv} be equal to 1 if node u is connected to node

[a]Recall that from the discussion in Section I, we must ensure $\cap_{i=1}^{L} C_i \subseteq C$

v and 0 otherwise. The following composition rules must be considered in the definition of C_{p_2}:

$$e_{uv} = 0 \qquad \forall u, v \in G \tag{10}$$
$$e_{uv} = 0 \qquad \forall u, v \in L \tag{11}$$
$$e_{u_1 v} + e_{u_2 v} \leq 1 \qquad \forall u_1, u_2 \in G, u_1 \neq u_2, \forall v \in B \tag{12}$$
$$e_{uv} = 0 \qquad \forall u \in G, v \in L \tag{13}$$
$$e_{uv} = 0 \qquad \forall u \in L_{DC}, v \in B_{AC} \tag{14}$$
$$e_{uv} = 0 \qquad \forall u \in L_{AC}, v \in B_{DC} \tag{15}$$
$$e_{uv} = 0 \qquad \forall u \in G, v \in B_{DC} \tag{16}$$
$$\tag{17}$$

These constraints impose that generators cannot be connected to generator; loads cannot be connected to loads; generators cannot be connected directly on the same bus; generators cannot be directly connected to loads; DC loads cannot be connected to AC buses; AC loads cannot be connected to DC buses; and generators cannot be connected to DC buses.

To define the implementation constraints C_{m_2} we introduce the notion of a path in the power distribution system. Consider a set of connectivity requirements $F \subseteq \{(i, j) \in L \times G | y_{ij} = 1\}$ between generators and loads. For a requirement (i, j), let π_{uvij} be a binary variable that is equal to 1 if the path from i to j uses the connection from u to v. Obviously, the following must hold: $\pi_{uvij} \leq e_{uv}$, $\forall u, v \in V, y_{ij} \in F$. A unique path exists between generator j and load i if and only if the following conditions are satisfied:

$$\sum_{v \in V} \pi_{jvij} = 1 \tag{18}$$
$$\sum_{v \in V} \pi_{uiij} = -1 \tag{19}$$
$$\sum_{u \in V} \pi_{uvij} = \sum_{u \in V} \pi_{vuij} \tag{20}$$

The reliability provided by a path must satisfy the following constraint:

$$\sum_{u,v \in B} (\ln a_{uv} + \ln a_u) \pi_{uvij} \geq a_{ij} \tag{21}$$

where a_{uv} is the availability of a connector (e.g. a TRU, power converter, contactor), and a_u is the availability of a bus.

The cost function is a multi-objective function that takes into account the weight and the inefficiency of the power distribution system. Both these functions depend on the set $E = \{e_{u,v}\}$ of connectors instantiated in the architecture, the number of buses used by the power distribution system and and the number of buses crossed by paths from source to destination. Thus, an optimization algorithm that solves this optimization problem will provide an architecture with the least amount of buses and connections, and with the shortest path possible. This is no surprise and it is in accordance with standard architecture where power distribution systems are organized into a two level hierarchy.

However, in this formulation, we have not considered the role of failures and the fact that not all paths are active at the same time. In fact, the result of the synthesis problem from Step 1, may require the same load to be powered by more than one generator to satisfy reliability constraints. This set of generators are not connected to the load at the same time, otherwise they would be also connected to each other violating on of the constraints of our platform. For this reason, the power distribution synthesis step must be preceded by a partitioning algorithm that generates sub-sets of the connectivity requirements $Y = \{y_{ij}\}$ from Step 1 under fault conditions. This problem can be cast into a bin packing problem that aims at generating one sub-set $Y_F \subset Y$ for each fault condition such that all loads are powered and generator efficiency is maximized. The power distribution system design can then be formulated as an optimization problem with the additional constraint that for each pair of generators, the paths departing from them be disjoint. This condition will guarantee that a contactor configuration can be found so that generators never share the same bus at the same time. The result of Step 2 can then be used to synthesize a state machine that handles power transfers of the electric power system.

VI. Conclusions and future work

In this article we presented a formalization of the design exploration activity for complex systems in the context of the platform-based design methodology. The methodology is general and the advantages are numerous as it allows correct-by-construction design, thereby reducing the verification effort, and allows to explore large design spaces to improve optimality. However, the major challenge to overcome for a successful adoption of the methodology is the articulation of the design flow into refinement steps such that the complexity of the design exploration problem is contained while maintaining the optimality of the result. These process also requires to understand the structure of the problem and build abstractions of the system components to be exported at the highest level of the the design flow to make informed decisions in the early stage of the design.

We used this driving principles in setting up a design flow for aircraft electric power systems. We present two refinement steps: generator selection and topology design. For these two steps, we also formulated the synthesis problem together with ways of dealing with their complexity.

We plan to extend our work in two directions. First, we plan to include storage elements in our library. These elements can be used not only to guarantee safety, but also to store energy that may be regenerated by actuators. Second, we plan to expand the approach to capture behavioral properties of the system such as power quality. This second extension include the automatic synthesis of discrete controllers used to command switches in the system, as well as continuous controller to guarantee the required power quality on each of the buses.

Acknowledgments

The authors would like to thank Mr. Paul Eremenko of DARPA's TTO office for funding this research. This work was performed under contract FA9550-07-C-0024.

References

[1]Sangiovanni Vincentelli, A., Carloni, L., De Bernardinis, F., and Sgroi, M., "Benefits and Challenges for Platform-Based Design," *Proceedings of DAC*, June 2004, pp. 409–414.

[2]Moir, I. and Seabridge, A. G., *Aircraft Systems: Mechanical, Electrical and Avionics Subsystems Integration*, AIAA Education Series, 2008.

[3]IBM, "Rational DOORS," .

[4]Steward, D. V., "The Design Structure System: A Method for Managing the Design of Complex Systems," *IEEE Transactions on Engineering Management*, Vol. 28, No. 3, 1981, pp. 71–74.

[5]Browning, T. R., "Applying the design structure matrix to system decomposition and integration problems: a review and new directions," *Engineering Management, IEEE Transactions on*, Vol. 48, No. 3, August 2002, pp. 292–306.

[6]Simmons, W. L., *A framework for decision support in systems architecting*, Ph.D. thesis, Massachusetts Institute of Technology, 2008.

[7]Pinto, A., Bonivento, A., Vincentelli, A. S., Passerone, R., and Sgroi, M., "System-Level Design Paradigms: Platform-Based Design and Communication Synthesis," *ACM Trans. on Embedded Computing Systems*, Vol. 5, No. 5, May 2006.

[8]Pinto, A., Carloni, L. P., and Vincentelli, A. L. S., "A Methodology for Constraint-Driven Synthesis of On-Chip Communications," *IEEE Transactions on Computer Aided Design*, Vol. 29, No. 3, March 2009.

[9]Balarin, F., Watanabe, Y., Hsieh, H., Lavagno, L., Passerone, C., and Sangiovanni-Vincentelli, A., "Metropolis: An Integrated Electronic System Design Environment," *Computer*, Vol. 36, 2003, pp. 45–52.

[10]Davare, A., Densmore, D., Meyerowitz, T., Pinto, A., Sangiovanni-Vincentelli, A., Yang, G., Zeng, H., and Zhu, Q., "A Next-Generation Design Framework for Platform-Based Design," *Conference on Using Hardware Design and Verification Languages (DVCon)*, 2007.

[11]Alexander, P., *System Level Design with Rosetta (Systems on Silicon)*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2006.

[12]Aerospace, S., *Architecture Analysis and Design Language (AADL)*, SAE, January 2009.

[13]OMG, *OMG SysML v. 1.1*, November 2008.

[14]Weilkiens, T., *Systems Engineering with SysML/UML: Modeling, Analysis, Design*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2008.

F.4 Assessing Performance Uncertainty in Complex Hybrid Systems

Assessing performance uncertainty in complex hybrid systems

Ritesh Khire[1], Sandor Becz[2], Hayden Reeve[3], Larry Zeidner[4]
United Technologies Research Center, East Hartford, CT, 06108

Complex systems are often a derivative of the current trend that dictates increased functionality and performance. Mobile phones that provide internet access and music players, fighter aircraft with fly-by-wire flight controls, and energy grids that monitor usage and recommend corrective actions are a few of the examples of emerging complex systems. There are a large number of causes that can lead to increased complexity of a system, such as (1) increase in number of parts, (2) increase in number of interactions, and (3) integration of multiple technologies. An increase in complexity makes these systems significantly challenging in terms of design and optimization, resulting in large time and resource investments. Therefore, it is typical for complex systems to use non-mature technologies that are expected to mature along the way. However, such infusion introduces significant levels of uncertainty, which historically have led to an explosion in development cost. In this paper, we assess the impact of uncertainty on a complex system. Through numerical simulations, we demonstrate that increasing complexity does not necessarily result in vulnerability of a system to uncertainties. These results challenge the conviction held by complex system design community, which assumes complexity and uncertainty have a direct correlation. Our results indicate that selection of appropriate system architectures is critical for system robustness. This should provide motivation for further research in the complex systems arena.

I. Introduction

The past several decades have seen the infusion of significantly new technologies in a wide range of systems, from cell phones to aerospace platforms. For example, the 787 and F-35 aircraft have incorporated more-electric systems for functions such as cabin pressurization and flight control actuation. This technology infusion has led to significant system architectural changes in an effort to improve the performance and capability of new platforms. As an example, (1) low maintenance cost and higher dispatch reliability requirements have led to the adaptation of prognostics and health management (PHM) systems, and (2) the implementation of fly-by-wire for flight control has reduced weight and pilot work load on many platforms. However, these and other driving forces have lead to an exponential growth in the complexity of modern aerospace platforms and a host of design and development challenges. In this paper, we are interested in assessing the impact of performance uncertainty on complex system design. In particular, we are interested in understanding the impact of uncertainty resulting from the infusion of different TRL (technology readiness level) technologies.

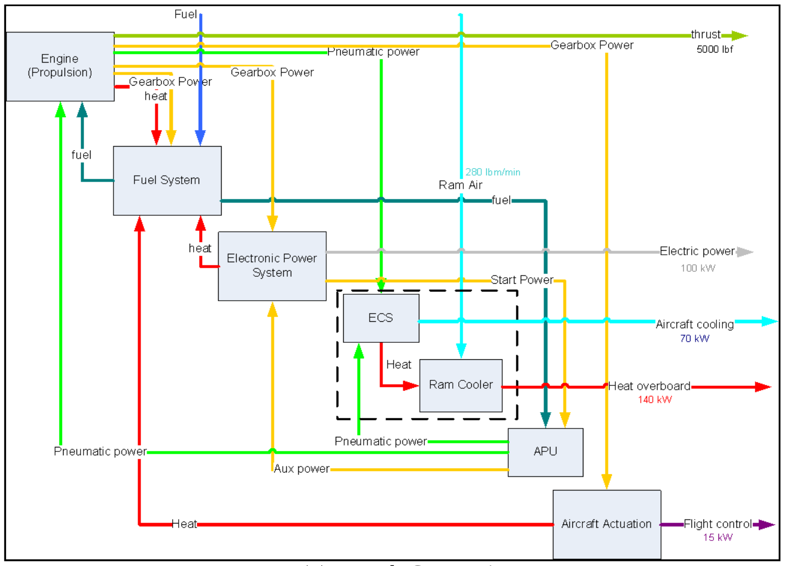
We note that throughout the paper, aerospace systems are used as a vehicle to discuss various issues involved with complex systems and also as an illustrative example. However, it is our tenet that the findings reported in this paper should be equally applicable to other complex systems, although they are not explicitly discussed.

A. Complex System -- Design Challenges

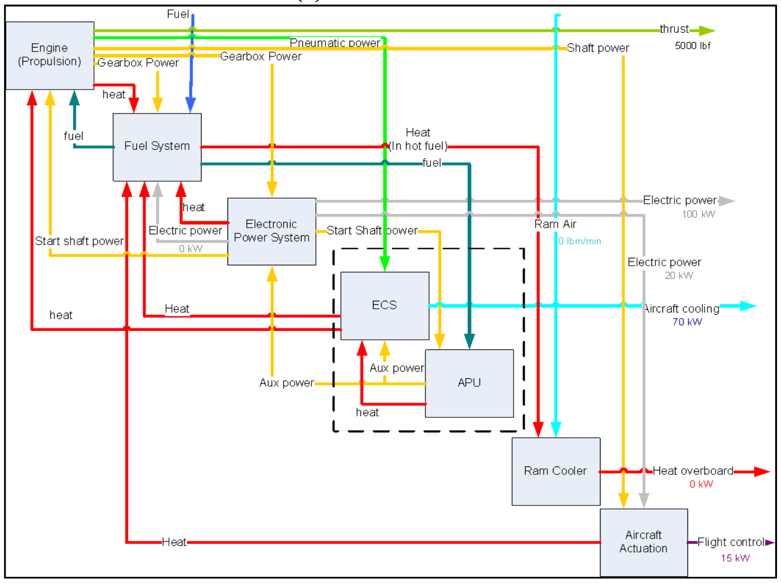
Continuing the system complexity discussion, the fifth generation tactical fighter aircraft offers a good example of the increased capability these new systems provide and also the resulting challenges. The fifth generation aircraft

[1] Sr. Research Engineer, Systems, 401 Silver Ln, East Hartford, CT 06108, and AIAA Senior Member.
[2] Staff Engineer, Thermal Mgt, 401 Silver Ln, East Hartford, CT 06108, and Member.
[3] Staff Engineer, Thermal Mgt, 401 Silver Ln, East Hartford, CT 06108, and Member.
[4] Staff Engineer, Thermal Mgt, 401 Silver Ln, East Hartford, CT 06108, and Member.

is the only one capable of transitioning from vertical flight to supersonic cruise. The incorporation of internal stores and a composite airframe provide superior survivability. Its design meets multi-role objectives through three variants for Short Take Off and Vertical Landing (STOVL), aircraft carrier operations, and conventional operations. In short, it offers 3 to 8 time the operational capability of fourth generation aircraft. This has come at the cost of increased system complexity. Fifth generation aircraft have approximately 130 subsystems, order 10^5 interfaces, and 90 percent of its functions managed by software [1]. This is a substantial growth from fourth generation aircraft that have approximately 15 subsystems, order 10^3 interfaces, and 40 percent of its functions managed by software [1]. It is interesting to note that a two order magnitude increase in interfaces (interactions) resulted from a single order magnitude change in the number of sub-systems. Figure 1 shows a comparison between the electrical systems of fourth and fifth generation aircraft. The reader should note the increase in the number of interactions.



(a) Fourth Generation



(b) Fifth Generation

Figure 1: Comparison of electric system architecture between fourth and fifth generation aircraft

The large number of sub-systems and interactions involved in a complex system poses significant design and evaluation challenges. As a result, development of complex systems is marked by large time and resource investments. Given this behavior, it is not unusual for complex systems to introduce low TRL technologies that are expected to be mature at the time of product launch. The introduction of composite primary structural members in next-generation passenger aircraft is a classic example of this phenomenon.

B. Sources of Uncertainty -- Technology Maturity

The introduction of immature or new technologies introduces significant uncertainties in the development of complex systems. This uncertainty is manifested in the form of a lack of accurate characterization of the subsystems during the early design and selection phase. This is in addition to other well known sources of uncertainty such as environmental conditions, evolving system requirements, etc. With the variability and uncertainty associated with parameters in complex systems, a formal treatment of their impact on emergent behavior must be included in any new design paradigm. This component is virtually non-existent in current design systems, particularly in early phase design.

In this paper, we assess the impact of uncertainties on complex system selection. Through numerical simulation, we demonstrate that the selection of good system architectures is critical to minimize the vulnerability of complex system to the above mentioned uncertainties. In other words, selection of appropriate complex system architectures will allow all functional requirements to be fulfilled. At the same time, the system will be robust against uncertainties, potentially reducing development time and resource investment.

On the contrary, a non-robust system architecture would result in the inability of the design to meet increased performance requirements while still meeting cost and schedule limitations. To illustrate the issue, consider next generation aircraft design. The introduction of more-electric and composite aircraft technologies (low TRL) has greatly increased the thermal management challenge on emerging and future platforms. The move to more-electric systems has increased aircraft heat loads (avionics, power electronics, and generator heat loads). Simultaneously, composite structures and low observable requirements have reduced the ability to reject heat from the aircraft. Furthermore, these platforms introduce highly integrated and technology rich subsystems. The end result is a thermal management system that may greatly limit aircraft performance and mission operability. Unless performance uncertainties in more-electric and composite technologies are taken into consideration, the increase in subsystem complexity and coupling can lead to significant development risks and increased development time/cost.

C. Impact of Technology Uncertainty on Development Cost

Analysis conducted by the US Government Accountability Office (GAO) of major defense acquisition programs found that research and development costs are 42% higher than originally estimated and that the average delay in delivering initial capability to the war-fighter is 22 months [1]. Analysis by the RAND Corporation found that the largest component in the growth in the cost of fixed wing aircraft has come from increased complexity [2]. As systems have become more complex they have become not only more expensive to develop, but the ability to predict that development cost is not accurate. Managing and minimizing complexity of new system development offers the ability to reduce both the magnitude and unpredictability of development cost. The GAO found that development programs that had more knowledge earlier in the development cycle incurred reduced cost overruns. Specifically: programs that start development with fully mature critical technologies experienced 30 less R&D cost growth; programs that held system engineering reviews (requirements review, functional review, or preliminary design review) prior to development start experienced 20% less cost growth; finally, programs that had no changes in key performance parameter requirement had three times less cost growth than other development programs. This statistics suggests that the impact of technology maturity (and uncertainty ensued by it) can be significant in terms of development cost. Additionally, it also provides a motivation for evaluating impact of uncertainties in the early stages of complex system design process.

D. Scope of Study

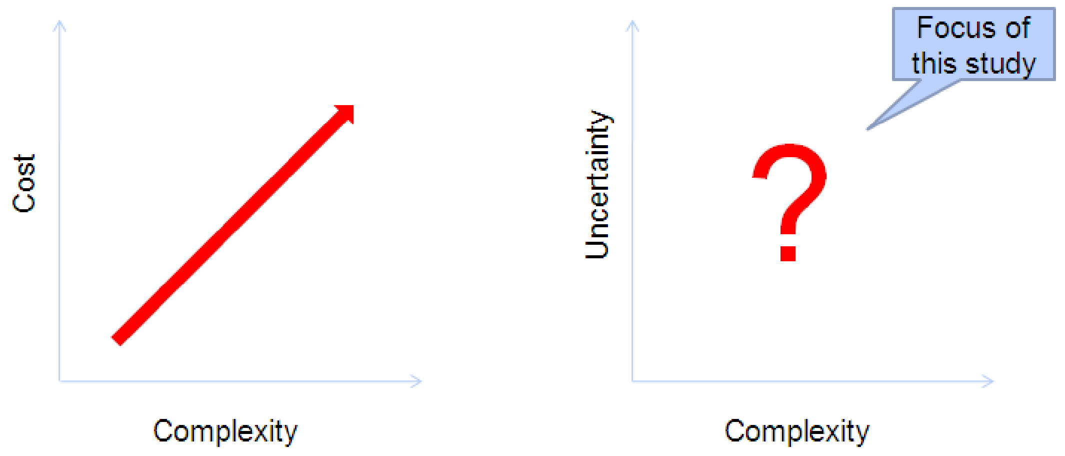


Figure 2: Motivation and Scope

From the above discussion, we observe that complex system development typically involves significant cost and time investment. It is important to note that the exact relation between complexity and development cost is yet to be accurately determined by the research community. However, it is typical for the development cost to be somewhat correlated to the complexity of the system. This is notionally reflected on the left side of Error! Reference source not found..

We also observed that uncertainty resulting from technology infusion plays a significant role in development cost. What is not clear from the above discussion is the relationship between systems complexity and uncertainty, as shown in Figure 2. In this paper, we focus on uncovering the role of uncertainty in complex system behavior. Ultimately, we are interested in identifying key drivers of uncertainty that dictate complex system behavior. Such information will be extremely valuable in the design of complex systems, as it will ensure that huge development cost over-runs are avoided or minimized.

II. Literature Survey

In this section, we provide a brief literature survey, which is expected to motivate the reader about the research gap. In the final manuscript, we will provide more exhaustively survey of the literature related to the role of uncertainty in complex system.

We divide the literature survey into two sub-sections. In the first subsection, we discuss the state-of-the art in terms of uncertainty assessment of hybrid engineering systems. These systems are of relevance and primary importance to us. This is followed by discussion on non-engineering systems.

A. Hybrid engineering systems

There has been considerable work in defining complexity that can guide design decisions. Kim and Wilecon [3] provide a review of complexity definitions that have been developed. These definitions cover the range of project, product, R&D/innovation, integration, and market areas and the definitions include the following core elements: (1) Number of parts, technologies, or functions required in a product, (2) Level of interdependency, (3) Novelty of project, and (4) Limitations such as time to market, requirements. Other definitions have been used in the aerospace field. AFRL's INVENT program [4] views complexity as equivalent to the inflexibility of a design to meet future growth requirements. That is, how tightly integrated the design space is with respect to change. Arena et al.[2] used the term complexity loosely to refer to the increased capability of aircraft. Jones et al.[5] developed a quantitative metric to estimate the cost of large scale systems by developing a metric based on the number of nodes and links within a system. What is missing from the literature is the handling of uncertainty in the context of complex systems.

B. Non-engineering Systems

System complexity metrics have a long history in the context of software quality control [6]. Typical metrics - such as McCabe Cyclomatic Complexity Metric [7, 8] of system complexity, use the graph structure of the underlying system in order to formulate a metric. There are a number of different studies, some of them in large industrial concerns such as IBM and Hughes Aircraft [9], that attempt to correlate the metric with the number of errors (and thus decreased productivity) in the code, with mixed results. Shepperd and Ince [8] analyzed the three most widely used software complexity metrics and identified that each was poorly formulated, involved implicit assumptions that were not characteristic of the ways in which the metrics were being applied, and had misleading or faulty statistical validation. One of the limitations of the complexity measures for software systems (which are largely homogenous in nature) is their applicability to the hybrid nature of engineering systems. Also, the level of uncertainty incorporated in terms of maturity is not as significant as that in engineering system, given the shorter development cycle for software system.

III. Evaluation of Uncertainty in Complex Hybrid systems

A. Problem Definition

We use electric power system of an aircraft as an example to evaluate the impact of uncertainty on complex system. In this illustrative example, the system is required to satisfy the requirements listed in Table 1.

Table 1: Requirements for electric system

System Requirement	Quantity at Cruise
Engine thrust	5000 lbf
Electrical power	100 kW
Cooling load	70 kW
Actuation power	10kW
Heat Sink capacity of engine	500kW
Heat Sink capacity of fuel system	Proportional to flow rate

An electrical system typically consists of a number of sub-systems, which are shown in the first column of Table 2. The second column shows various technology options considered for each sub-systems. As we shall see later in the paper, multiple system architectures can be developed by selecting (1) alternate technology choices and (2) sub-system interactions. Generation of different system architectures is governed by different combination of sub-system functionalities such that requirements specified in Table 1 can be satisfied. These functionalities are listed in the third column of Table 2. We shall also see that every system architecture will have a complexity number associated with it. In this study, the complexity of system architecture is quantified based on the number of sub-systems and the number of interactions between them. Finally, to evaluate the effect of uncertainty on complex system, we will introduce uncertainty in each sub-system. This uncertainty is expected to simulate performance unpredictability of different technology options. As discussed earlier, the level of performance unpredictability is related to the technology maturity. The final column of Table 2 shows the uncertain parameter considered in this study.

Table 2: Technology choices available for aircraft sub-systems

Sub-System	Technology options	Functions^{+++}	Uncertain parameter
Engine	N/A	Provide thrust	Thrust specific fuel consumption (TSFC)
		Provide shaft power	
		Provide pneumatic power	
		Heat sink and source	
Fuel System	Shaft power input	Supply metered fuel	Efficiency
	Electrical power input	Heat sink and source	
EPS- Electric power	Shaft power input	Supply electric power	Efficiency
		Heat source	
ECS-Cooling Systems	a) Electrical power input	Provide cooling	COP
	b) Pneumatic power input	Heat source	
	1. Vapor compression cycle		
	2. Air compression cycle		
	3. Heat rejection to engine		
Ram Cooler	Ram air operated	Heat sink	Efficiency
Actuation	Electrical power input	Provide flight control actuation	Efficiency
	Shaft power input	Heat source	
	Combined electrical-shaft		

B. Uncertaity evaluation process

Figure 3 shows the process that we used to evaluate the impact of uncertainty. As shown in Figure 3, we first generate multiple architectures that have the potential to satisfy requirements. Next, a physics based model is developed for each architecture by combining the models of individual sub-systems. This model is used to evaluate the architecture against requirements specified in Table 1. Also, the same model is used to run Monte Carlo simulation against the uncertain parameters described in Table 2. The Monte-Carlo simulation provides the probability-of-feasibility for satisfying requirements. Once all the potential architectures are evaluated, they are compared in terms of complexity and probability-of-feasibility, and an architecture is selected that offers best compromise between the two.

In the next few sub-sections, we discuss key processes.

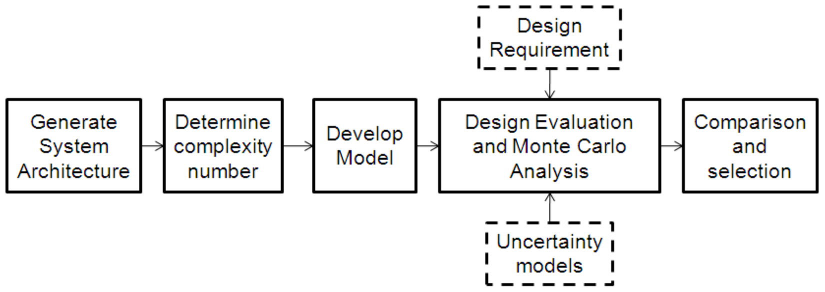

Design Requirement
Generate System Architecture
Determine complexity number
Develop Model
Design Evaluation and Monte Carlo Analysis
Comparison and selection
Uncertainty models

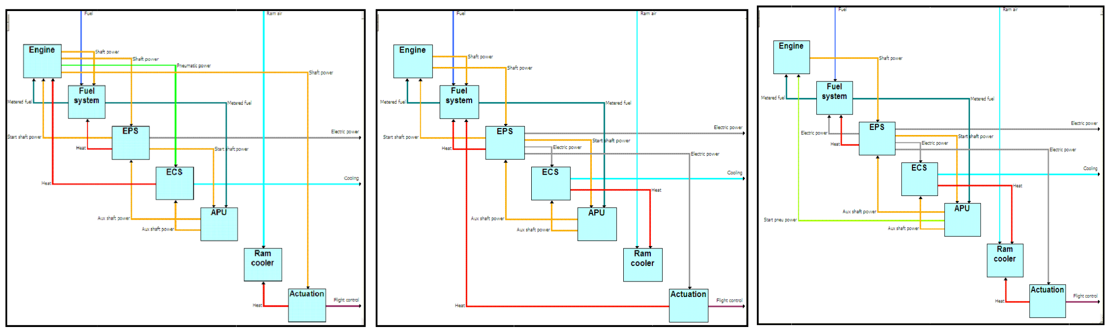
Figure 3: System architecture uncertainty evaluation process

C. System Architecture Generation

As discussed before, we can generate multiple system architectures by selecting different technologies and/or by changing interactions between various sub-systems. We have developed an Architecture Enumeration and Evaluation (AEE) tool [10] that exhaustively combines different sub-systems and automatically generates all potential architectures. The AEE tool uses two criteria to filter infeasible architectures: (1) *potential* to fulfill design requirement and, (2) complete functionality matching. Since the discussion of AEE development is beyond the scope of the current paper, we will discuss its developmental details in a separate publication. Here, we only discuss the above two criteria and the architectures that are used evaluate impact of uncertainty.

Figure 4 shows three candidate architectures generated in this study. We refer to architectures shown in Figure 4 (a), (b), and (c) as all mechanical, partial electric, and all electric, respectively. In the all mechanical architecture, engine drives fuel system, EPS, ECS, and Actuator sub-systems by providing direct shaft power. Whereas in the all-electric architecture; engine drives EPS, which drives other unit using the generated electrical energy. In the case of partial electric architecture, engine and EPS share the responsibility of other sub-system operations.

From the above three architectures, we can see that the engine has different number of sub-system interactions with other sub-systems. Also, we can see that the actuation subsystem can be electrically operated or mechanically operated, which represent two distinct technology choices. Similar observations can be made about other sub-systems shown in Figure 4. We note that the architectures shown in Figure 4 have the *potential* to satisfy all the requirements specified in Table 1. In other words, they are all capable of supplying thrust, electric power, cooling load, and actuation power.



(a) All Mechanical *(b) Partial Electric* *(c) All Electric*

Figure 4: Different Aircraft Architectures

From Table 2, we can observe that a number of sub-systems act as heat source or heat sink (i.e. they can either generate heat or can absorb it). To ensure that all the generated heat is dissipated, each heat source should pair with at least one heat sink. Without this functionality matching, the architecture may result in over-heating and failure of aircraft system. Similar function matching can be sees in terms of energy source and sink in the three architectures shown in Figure 4.

D. Complexity measurement of an architecture

There are many principles that describe how elements of complexity propagate and drive system cost. These principles can be combined to form the theory for a complexity metric. Examples of complexity principles include:

- Node complexity: a system’s complexity increases as the complexity of its constituent parts increases
- Number of nodes: the complexity increases as the number of constituent parts increases
- Number of links: the complexity increases as the number of interconnections increases
- Types of links: the complexity that each type of interconnection adds to a system is based on the type of energy/fuel/data flow
- Roll-up: a module encapsulates the complexity of the elements inside it and their interconnections with one another and with the module’s environment
- Shared conveyance: multiple interconnections of the same energy/fuel/data flow type linking two embedded modules may present lower complexity together than separately
- Pass-through: an interconnection passing from one embedded subsystem to another represents complexity within those embedded systems, but little complexity in their parent system, which adds minimal value to the flow
- Complexity isolation: clustering nodes and their interconnections into hierarchical modular clusters localizes complexity so that other modules and higher levels may have lower complexity
- Clustering cost: while clustering is beneficial, it is balanced in the complexity metric by an associated cost

Complexity principles can be embedded in a complexity metric. For example, Jones, et al. [5] present a complexity metric that embodies a few of these principles: node complexity, number of nodes, and number of links. Their complexity metric is:

$$RDT\&E\$ = aN_e^b \quad \text{where}$$

$$N_e = \underbrace{\left(aN_{s/r} + bN_s + cN_r\right)}_{\text{Capture complexity associated with type of Nodes}} \underbrace{\left(\frac{L_t/N_t}{\text{avg}\left(L_t/N_t\right)}\right)^d}_{\text{Capture the connectivity complexity of the system}}$$

The node complexity principle is embodied in the distinction between three types of nodes: receivers (Nr), transmitters (Ns), and transceivers (Ns/r), and their multiplication by three separate complexity factors, h, g and d. The number of nodes principle is embodied in multiplication by the term (dNs/r + gNs + hNr). The number of links principle is embodied by the term Lt. We note that the metric presented by Jones does not embody the types of links principle. However, we can adapt it to embody this principle by changing the interpretation of Lt to represent the sum of the complexity of each link, based on energy type.

In the current study, we implement the above complexity metric

E. Architecture Model Development

As stated previously, we develop a physics based model for each architecture. The architecture level physics based model primarily consists of models of individual sub-systems. In the aircraft electric system example, we used parameterized, low-order, steady-state models for each subsystem. These low order models were deemed sufficient (conservative to cover non-steady state effects) to evaluate each subsystem. In the final paper, we will provide more information on the details of these models.

F. Uncertainty Analysis -- Monte Carlo Simulation

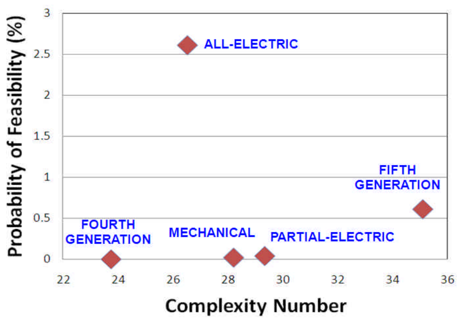
Once the architecture model is developed, it is simulated against the requirements specified in Table 1 and uncertain parameters specified in Table 2. As noted before, the uncertain parameters represent the uncertainty in predicting the performance of individual sub-systems. We use Monte-Carlo technique to evaluate effect of uncertainty on complex system architectures. The architectures that do not satisfy the heat-sink requirements specified in Table 1 are deemed infeasible.

Next, we discuss the results of uncertainty analysis on the complex system architectures considered in this study.

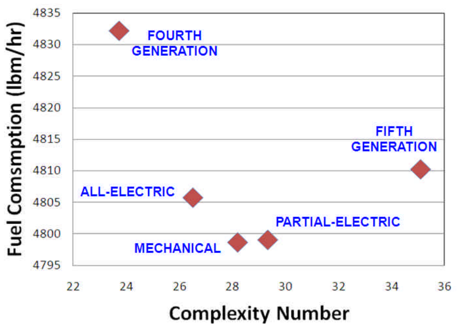
IV. Discussion

A. Complexity Vs Uncertainty

Figure 5 (a) shows the comparison between system complexity and probability of infeasibility for architectures evaluated in this study. The first and important observation from Figure 5 (a) is that probability of infeasibility is not proportional to the complexity of the architecture. In other words, a more complex system is necessarily more vulnerable to the uncertainty associated with technology maturity. Instead, the selection of system architecture is critical for minimizing risk associated with uncertainties.



(a) Complexity vs Feasibility



(b) Complexity vs Fuel consumption

Figure 5: Relation between Complexity, Feasibility, and fuel consumption

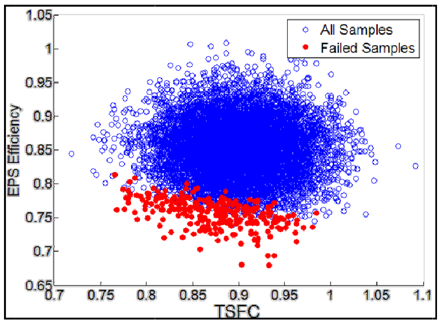
Among the architectures evaluated in Figure 5 (a), all-electric architecture appears to be most vulnerable, followed by the fifth generation and partial-electric architectures, respectively. From Figure 1 and Figure 4, we can observe that these three architectures rely on varying levels of electrical energy usage, starting from highest to lowest, respectively. On the other hand, mechanical and fourth generation architecture relies on the least amount of electrical energy (EPS is not required to power other sub-systems).

For the five architectures studied so far, the trend suggests that inappropriate/blind infusion of electrically operated technologies in aircraft architectures can lead to significant levels of risks. One can minimize the levels of risks by selecting partial electric architecture, where all subsystems except fuel system are electrically operated. However, from Figure 5 (a), partial architecture is more complex than all-electric one. The increased complexity is due to the mechanical interface (shaft + bearings + lubrication) between fuel system and engine in partial architecture, as oppose to electrical interface (wire) between fuel system and EPS in all-electric architecture. From Figure 5 (b), we can observe that all the architectures have similar fuel consumption (within +/-1%, which is within tolerance for low order model). So, selection of any architecture is not expected to significantly alter fuel economy.

It is important to note that, electrically operated technologies are merely illustrative in this case study. Broad inference is that the infusion of technologies in complex architecture needs to be done with utmost care. It is possible to infuse new technology, provided an appropriate architecture is selected. Designer can select an appropriate architecture by following the process described in Figure 3.

B. Parametric Evaluation of Infeasibilities

To better understand how uncertainty manifests into an infeasible architecture, we study the effect of various efficiency distributions. For the all-electric architecture, Figure 6 shows all the Monte-Carlo samples on TSFC vs EPS Efficiency and TSFC vs Fuel System Efficiency plots. The sample points that resulted in infeasible designs are shown in red, and the feasible points are shown in blue.



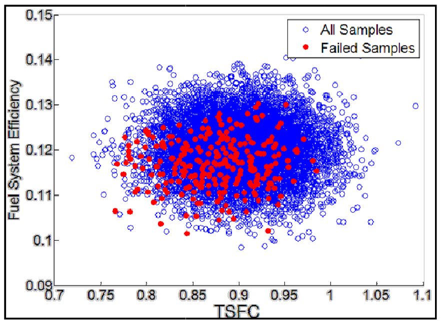


Figure 6: Effect of subsystem efficiency on feasibility of all-electric architecture

From Figure 6, all infeasible points are concentrated in a low EPS efficiency region. On the other hand, for TSFC and fuel system efficiency, the infeasible points are not concentrated into a particular region. This observation indicates that EPS efficiency is a critical parameter in the all-electric architecture, which appears intuitive. However, further investigation into results reveals that all the infeasibilities are due to the heat-sink capability of the fuel system. In other words, the current fuel system is not able to dissipate the higher amount of heat being generated at low EPS efficiencies. An easy fix would be to improve the heat dissipation capability of the fuel system.

It is important to note that in the case of all-electric architecture, the cause of infeasibility was not necessarily intuitive. It was not necessarily the inability of the EPS system, but instead that of the fuel system. This observation illustrates the coupled nature of the complex system, which makes their evaluation a challenge. Also, evaluation of root cause of failure may be more involved, and would require more sophisticated methods than those based on linear sensitivity analysis

V. Concluding Remarks

Infusion of low maturity technology introduces significant source of uncertainty in the development of complex systems. Historical data shows that some of the large scale projects have resulted in significant budget overruns due to improper/immature infusion. In this paper, we have demonstrated that there is no clear relation between the complexity of the system and its susceptibility to uncertainty. The encouraging results from our study indicate that the selection of an appropriate architecture is critical to ensure that the resulting system is not vulnerable to uncertainties.

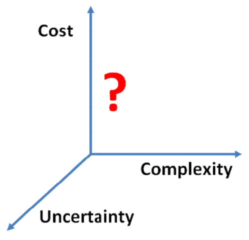


Figure 7: Desired outcome

In the final manuscript, we plan to include results from more architecture analysis. The objective of further simulations is to identify key complex system architecture parameters that can dictate its feasibility under uncertainties and development cost, as shown in Figure 7. Such information would be important to develop an automated exploration/optimization technique that will develop appropriate architecture.

VI. Acknowledgments

The development of the methods and results contained in this study was funded by Mr. Paul Eremenko of DARPA TTO under the following contract: DARPA Abstraction Based Complexity Management contract FA9550-07-C-0024.

VII. References

[1] http://www.gao.gov/new.items/d09326sp.pdf
[2] Arena, M. V., Younossi, O., Brancato, K., Blickstein, I., Grammich, C. A., "Why Has the Cost of Fixed-Wing Aircraft Risen?" RAND Corporation, 2008, http://www.rc.rand.org/pubs/monographs/2008/RAND_MG696.pdf
[3] Kim, J. and Wilemon, D. "An empirical investigation of complexity and its management in new product development", Technology Analysis & Strategic Management, 5/2009; 21/4: 547-564
[4] Walters, E. A., and Iden, S., "INVENT Modeling, Simulation, Analysis and Optimization", AIAA Aerospace Sciences Meeting, Orlando, FL, 2010-287.
[5] Jones, R., Hardin, P., and Irvine, A., "Simple Parametric Model for Estimating Development (RDT&E) Cost", 2009 ISPA/SCEA Joint Conference.
[6] Sangiovanni-Vincentelli, A., "Quo Vadis, SLD? Reasoning About The Trends and Challenges of System Level Design," Proceedings of the IEEE, 2007, Vol. 95, No. 3, pp. 467-506.
[7] McCabe, T.J., "A complexity measure. IEEE Transactions on Software Engineering," 1976. Vol. 2, No. 4, pp. 308–320.
[8] Shepperd, M. and Ince, D.C., "A critique of three software metrics," Journal of Systems and Software, 1994, Vol.26 , No. 33, pp. 197-210.
[9] Navlakha, J.K., "A Survey of System Complexity Metrics," The Computer Journal, 1987, Vol. 30, No. 3, pp. 233-238.
[10] Zeidner, L.E., St. Rock, B.E., Desai, N.A., Reeve, H.M., and Strauss, M.P., "Application of a Technology Screening Methodology for Rotorcraft Alternative Power Systems," *AIAA Aerospace Sciences Meeting*, AIAA-2010-1505, Orlando, FL 2010.

F.5 Architectural Enumeration

Architectural Enumeration & Evaluation for Identification of Low-Complexity Systems

Lawrence E. Zeidner[1], Sandor Becz[1] Hayden M. Reeve[1], and Ritesh Khire[1]
United Technologies Research Center, East Hartford, CT 06108, USA

The cost of large complex systems can be reduced by using system complexity as a cost proxy during the initial stages of system architecting. This paper presents Architectural Enumeration and Evaluation (AEE), a method for rapid, efficient and thorough consideration of enormous architectural design spaces to find the best low-complexity solutions. AEE assembles promising system architectures from any number of candidate technologies and evaluates them relative to a set of customer-value metrics, which can include a complexity metric as a proxy for cost. This paper also presents an example application of AEE together with a complexity metric and a spectral graph partitioning method, to enumerate the feasible set of architectures, and identify for each architecture its lowest complexity hierarchical clustering of subsystems.

Nomenclature

AFRL	=	Air Force Research Lab
INVENT	=	Integrated Vehicle Energy Technology
CCI	=	Capability/cost index

I. Introduction

Significant technological and architectural changes have been introduced into aerospace systems, over the past several decades, in an effort to improve the performance and capability of new platforms. These changes have led to an exponential growth in the complexity of modern aerospace platforms and accompanying design and development challenges. These increases in technical complexity have been accompanied by increases in the complexity of system requirements and organizational partnerships. Complexity in requirements stems from the need to meet multiple present and future mission requirements. Analysis by the RAND Corporation found that the largest component in the growth in the cost of fixed-wing aircraft has come from increased complexity. [7] As systems have become more complex their development cost has increased and the predictability of their development cost has decreased. Managing and minimizing complexity of new system development offers the ability to reduce the magnitude and increase the predictability of development cost. Fundamental architectural decisions made early in the design process have a major impact on system complexity; however, system architects lack quantitative feedback on the complexity of their architectures.

During the early design stages of large systems, systems are architected in terms of functional subsystems (e.g., Electrical Power System, Environmental Control System, Auxiliary Power Unit) and their interconnections as shown in Figure 1. This standard top-down design approach is a divide-and-conquer strategy that defines the sub-system boundaries as an interface to decouple the overall design problem into two sequential sub-problems. The first sub-problem is to design a system of functional subsystems based on overall system requirements, resulting in a set of requirements for each subsystem. The second sub-problem is to design each functional subsystem in terms of its required functionality. This approach has also enabled airframers to focus on the large-scale system-integration problem, while outsourcing subsystem design to subcontractors, with the subsystem requirements serving as the organizational interface. Since these two sub-problems are not really decoupled, subsystem interactions necessitate design iteration to meet system-level requirements. Because functional subsystems are designed separately, these iterations often occur late in the design process and are a significant cause of schedule and cost overruns.

[1] Research Engineer, Thermal Management, United Technologies Research Center, 411 Silver Lane, MS 129-89, East Hartford, CT, 06108.

One approach to design an overall system comprised of functional subsystems (the first sub-problem) is to enumerate and evaluate all feasible solutions to this problem, as described in [1]. This approach has typically been avoided due to the exponential size of the design space. However, it has recently been shown that the feasible set is actually quite sparse, many orders of magnitude smaller than the overall design space, and that generative filters can be used to identify the entire set of feasible solutions directly, rather than searching through the design space for feasible solutions. [1] Using this approach, each architecture in the feasible set is specified in terms of the type of information depicted in Figure 1.

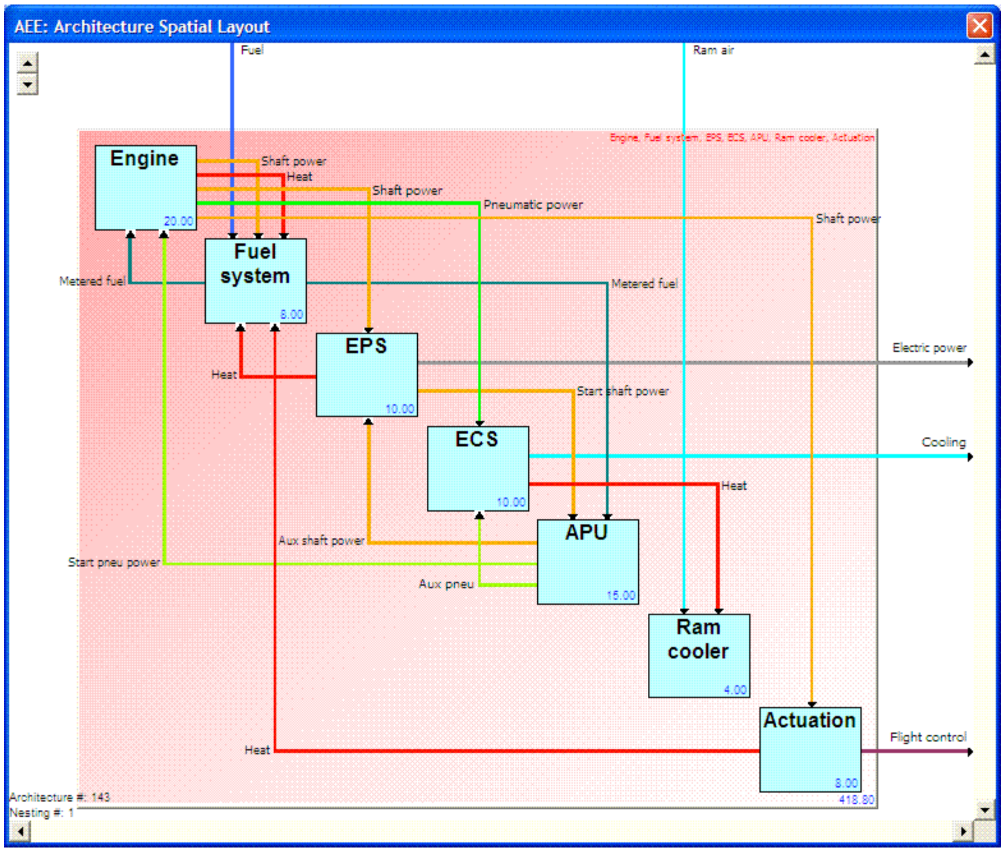


Figure 1 - An aerospace system architecture specified in terms of its functional subsystems and their interconnections, with a complexity metric assessed for each subsystem and for the overall system.

The complexity of each architecture depends not only upon its constituent technologies and its interconnections, but also on how and to what degree the architecture is organized hierarchically into modules. Each architecture can be decomposed into many different hierarchical configurations of modules, each with its own degree of complexity. [2] presents the design issues pertaining to the complexity metrics suitable for hierarchical system configurations, while [3] presents the use of spectral graph partitioning coupled with a complexity metric to find the lowest-complexity hierarchical clustering of the subsystems.

Complexity metrics can roll up the complexity within each hierarchical module based on the complexity of its internal elements and interconnections. The ability to quantify and thereby minimize complexity of advanced systems throughout the design process is required. This, in turn, requires both abstract (i.e. low fidelity) complexity metrics as a leading indicator of complexity for use early in the process during configuration selection, cost estimation, and bidding and proposing, all the way to detailed complexity metrics for use during detailed design. [2]

System modularity is a well-known technique for reducing system complexity by clustering system elements into modules to maximize connections within modules and minimize connections between them. [3] presents a method for identifying a hierarchical clustering of modules to reduce system complexity. This method is able to employ spectral graph partitioning based on an adjacency matrix that is constructed using information available early in the design process, and couples this with a complexity metric, by use of the WBestCut partitioning algorithm. [3]

The remaining sections of this paper are organized as follows. Sections II and III will define the problem and present a solution approach. Section IV will provide an example application of this solution approach. The paper concludes with a summary and with suggested extensions.

II. Problem Definition

This problem addresses large systems that consist of devices that embody technologies that convert, transfer, and store energy, information and/or material, and interconnections between those devices that enable energy, information and/or material flow. In general, these classes of devices span an extremely large and diverse set of technology options available for consideration. To further complicate the problem, the performance characteristics of existing technology is continually improving at different rates, and emerging disruptive technologies are changing the overall landscape. The large and constantly changing set of options makes the prediction of a global optimum system solution, with low complexity and thus low cost, extremely challenging.

To handle the scope of the problem within resource limitations, system engineers are usually forced to focus on a small subset of the overall design space, an approach that typically results in incremental improvements to existing system designs, and high cost. A more comprehensive consideration of the entire design space has the benefit of ensuring that all intuitive and non-intuitive architectures are considered, greatly increasing the likelihood that the global optimum low-complexity system solution is discovered early in the design cycle. Comprehensive design space exploration provides the additional benefits of 1) providing the quantitative information necessary for data-driven decision-making early in the design process, 2) enabling the measurement of technology gaps relative to application requirements, 3) quantifying performance targets required for the insertion of a particular technology into key applications, and 4) identifying the impact of new or competitive technologies on UTC applications.

The process of comprehensively considering all of the possible ways to interconnect a set of devices is called enumeration of the design space. Usually, enumeration is not used to solve complex system problems due to the extremely large set of possibilities that are produced.

The general problem can be expressed mathematically as a combinatorics problem of enumerating all graphs having up to k nodes (the maximum number of devices allowed within an assembled system), for which each node can take on n colors (the total number of candidate technologies considered), and in which there can be up to m directed links (the number of energy, data or material flow types) in each direction, between node pairs. The design space consists of the total number of enumerated graphs (ntotal), which we will refer to in this paper as system architectures is at most:

$$n_{TOTAL} = n^k \cdot 2^{m \cdot \binom{k}{2}}$$

Therefore, if an analysis is limited to 10 candidate technologies interconnected into a system with $\leq$ 4 devices and 4 energy types, enumeration of the design space would consider more than 100 billion possibilities. The vastness of this problem substantially exceeds available computational and financial resources.

As a practical matter, however, only a small number of these 100 billion systems are feasible solutions (i.e. physically meaningful) as illustrated in the yellow area of Figure 2. A feasible system needs to meet all specific requirements for a given application and requires that the distinct inputs and outputs of each device are compatible with all other elements of the system. In general, the size of the feasible solution set is many orders of magnitude smaller than the size of the design space. Additional design-space reduction can be achieved by realizing that within the feasible solution set, only a small percentage of system architectures can be considered “promising” in terms of system performance relative to value metrics, while others can be considered non-competitive (i.e. feasible, but poor performing). The Architectural Enumeration & Evaluation (AEE) process and toolset rapidly generate and identify this promising set of system architectures using efficient filtering methodologies and an abstracted analysis framework, thereby enabling consideration of the entire design space. Other methods that explore the design space vs. considering it comprehensively include design synthesis methods via design structure languages and metamodeling [4].

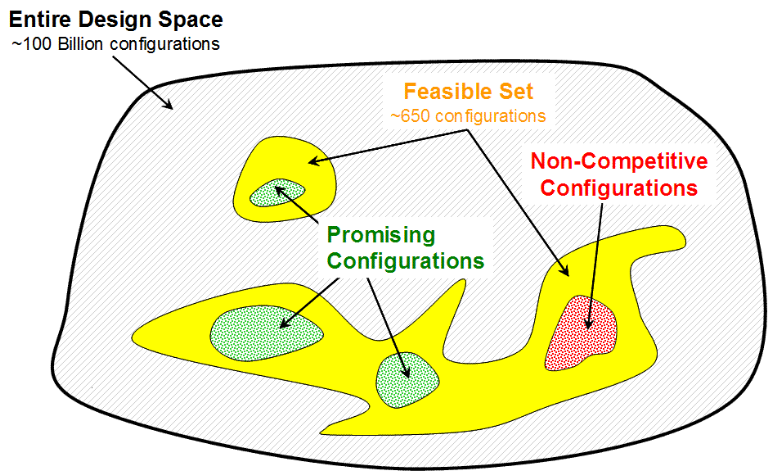


Figure 2 - Architectural design space

III. Solution Approach

Identification of the best system architecture, according to all relevant customer-value metrics, including complexity as a proxy for cost, presents two apparently conflicting demands. It requires a solution approach that rapidly spans an exceedingly broad design space, while providing high-fidelity analysis. Comprehensive consideration of the entire design space is needed to ensure that all intuitive and non-intuitive architectures are considered. High-fidelity analysis is needed to accurately distinguish among competing architectures. The challenge is to achieve comprehensive consideration as well as high-fidelity analysis in a timely fashion, since application of high-fidelity analysis to many architectures is time and cost prohibitive. The solution approach presented in [1] is a multi-level successive-refinement process called Technology Screening, illustrated in Figure 3. Here we show a two-level process, though the approach could be executed as any number of successive-refinement levels. Other successive-refinement approaches have been described in the literature, including platform-based design [5,6].

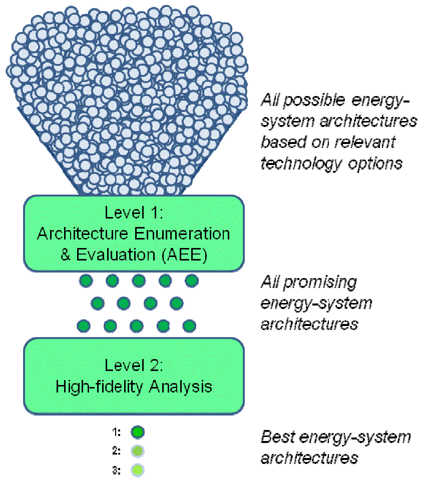

All possible energy-system architectures based on relevant technology options
Level 1: Architecture Enumeration & Evaluation (AEE)
All promising energy-system architectures
Level 2: High-fidelity Analysis
1:
2:
3:
Best energy-system architectures

Figure 3: Multi-Level Approach to Technology Screening

Within the two-level technology screening process shown in Figure 3, the first level, AEE, enables rapid and comprehensive screening of all possible energy-system architectures using a high-level of abstraction to enumerate

the design space, and a low-fidelity analysis to identify all promising architectures in the design space. Level 2 uses a low level of abstraction (i.e., high-fidelity modeling and optimization) to select the best concept(s) from among the promising architectures identified at Level 1 (Figure 4).
This solution approach is not limited to two levels. Either the AEE top level may be split into multiple successive-refinement levels, or the high-fidelity analysis may be split into multiple analysis and optimization levels, e.g., physics-based high-fidelity analysis, followed by higher-fidelity computational fluid dynamics analysis of flows, or detailed multi-physics analysis and optimization.

This multi-level approach enhances the efficiency of AEE and improves the quality of assessment because, at each successive level, the trade space is not only successively refined, but it can also be adaptively refined. This adaptive refinement can be conducted automatically. Issues absent from the previous higher level of abstraction can be introduced to the system model, based on which architectures are brought forward from the previous level as promising. For example, if architectures employing batteries are selected as the most promising architectures in Level n, then Level n+1 may include analysis of the battery chemistry, while if the architectures employing photovoltaics are the most promising in Level n, then Level n+1 may include analysis of the concentration ratio. For competitive technologies, adaptive refinement of technological attributes will improve the quality of assessment by enhancing fidelity for the issues which are most important to distinguish among the best system architectures.

Technology Screening enables the quantification of technology gaps for particular applications. If an emerging technology is not yet competitive for a particular application, it is useful to know how far its performance parameters would need to be improved for it to become competitive. Technology gap analysis is useful because it enables decisions to be made based upon the cost and time required to advance the technology's performance parameters enough for the technology to become competitive for a specific application. Gap analysis shows which performance parameters are most in need of improvement for a specific application, and whether there are multiple parameters needing improvement or only one. The identification of multiple performance parameters requiring improvement for a technology to become competitive avoids a common situation in which one parameter is improved, only to reveal the existence of a second performance parameter that must also be improved, introducing unforeseen delay and cost. Since the competitive landscape of technologies that can be focused on an application changes continually as technologies progress, the ability to update gap analyses easily and quickly is useful.

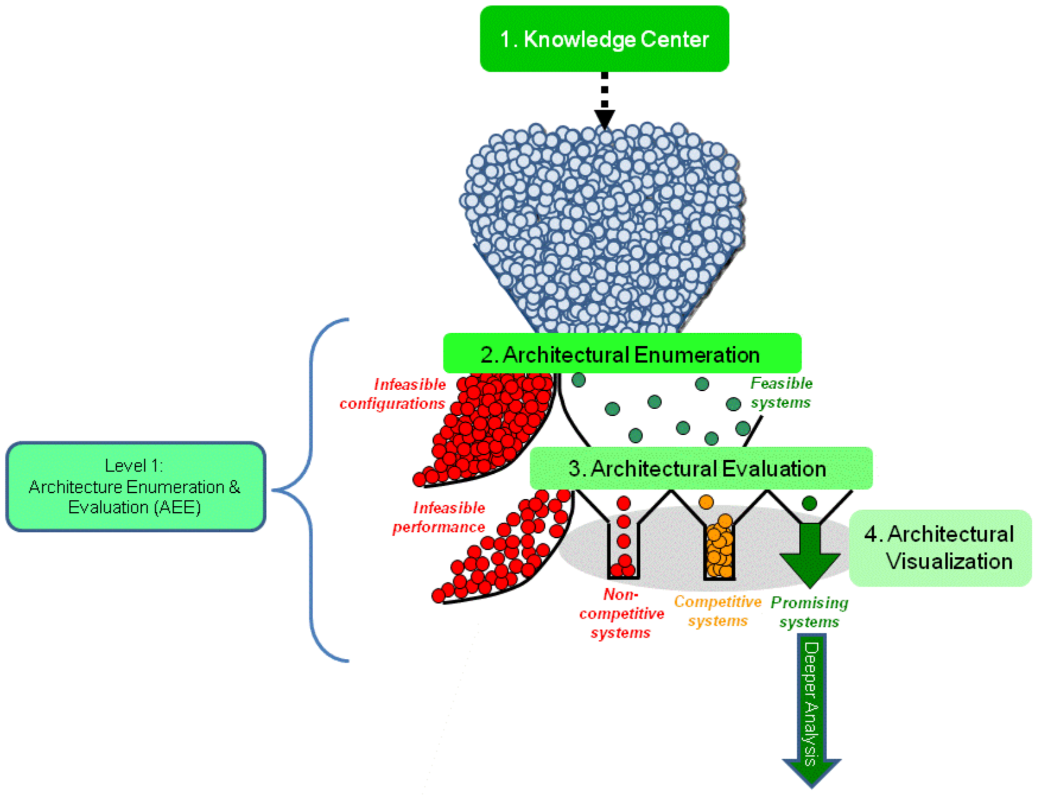

1. Knowledge Center
2. Architectural Enumeration
Infeasible configurations
Feasible systems
3. Architectural Evaluation
Infeasible performance
4. Architectural Visualization
Level 1: Architecture Enumeration & Evaluation (AEE)
Non-competitive systems
Competitive systems
Promising systems
Deeper Analysis

Figure 4: Sequential Application of the Four Elements of Technology Screening

The AEE process consist of four key elements, along with existing high-fidelity modeling methods, that provide the functionality needed to perform Technology Screening. These four elements are the sequential application of Architectural Enumeration, Evaluation and Visualization as sequential steps, once the Knowledge Center has been used to define the design space, as shown in Figure 4. This paper focuses on the AEE process within Technology Screening since various types of high-fidelity modeling (Level 2) are already well understood.

A. Knowledge Center

The Knowledge Center is an organization's central repository for tracking technology attributes both in the form of descriptions and models. It ensures a consistent basis for analysis and proposals, and is integrated with the other Technology Screening elements so that it is quick and convenient to use. The first step in Technology Screening is to use the Knowledge Center to determine and specify which technologies are relevant for the application.

Once Technology Screening has been used to determine technology gaps for specific applications, these gaps can be recorded in the Knowledge Center so that when a technology's performance parameters progress to close this gap, appropriate organizational work-flow messages can be generated. For example, if Technology Screening shows that current Li-ion battery technology has insufficient volumetric energy density for an aerospace application, then the technology gap can be quantified; the threshold of volumetric energy density required for Li-ion battery technology to be competitive for that aerospace application can be determined and recorded. This value can be updated whenever relevant inputs are changed. As technology performance parameters are being continually updated, when the level of Li-ion battery volumetric energy density nears the recorded threshold, the Knowledge Center can alert the relevant individuals within the organization about this change, so that they can be prepared to act quickly on this opportunity. With many applications and emerging technologies, this alert service can enable organizational agility in the face of accelerating change.

The Knowledge Center includes the following types of information:

1. Device Characterization

Each technology is embodied as a device along with qualitative descriptions and quantitative performance attributes and computational models to evaluate them. Qualitative descriptions of technologies include identification of fuel and energy types that are potential inputs and outputs for each technology, e.g., an electric motor accepts electric power as an input and produces shaft power as an output. Quantitative models for a technology may take the form of performance parameters, (e.g., power density, conversion efficiency or cost), a response function representing variation in a performance attribute as a function of known quantities, (e.g., conversion efficiency as a function of part load, temperature or altitude), or a table of data points for interpolation (e.g., empirical test data). Devices are organized hierarchically in the Knowledge Center by increasing level of specificity (e.g., a tree structure including a generic gas compressor at level n, positive displacement and dynamic compressors at level n+1, and as children of positive displacement compressors, reciprocating and rotary compressors at level n+2). Each level of device description inherits the attributes of its parent device, so that only those attributes that are different need specification.

This organization of device information by increasing level of specificity and this inheritance assumption simplify the task of adding a device to the Knowledge Center, simplify the task of selecting all relevant devices for a specific application, and provide a mechanism for automating adaptive refinement.

One approach to automating adaptive refinement is to leverage the hierarchical nature of the device tree in the Knowledge Center, so that the early levels begin with relatively generic devices, and then, based on which devices are involved in promising architectures, more specific devices can be substituted automatically at subsequent levels. For example, a generic heat engine may be assumed at Level n. If architectures including the generic heat engine are selected as most promising, the corresponding Level n+1 architectures may each have different types of heat engines substituted (e.g., Otto, compression, rotary, Brayton, …). This approach is adaptive in the way in which it applies refinement, because it uses the results of the previous level of refinement to determine where to apply the next level of refinement. For example, if architectures including the Otto engine are selected as most promising in Level n+1 as opposed to the other types of heat engines, the corresponding Level n+2 architectures may each have different types of Otto engines substituted (e.g., radial, V, inline, horizontal piston, opposed piston, …), while if the type of Otto engine was not a distinguishing feature at Level n+2 (all Otto engines performing roughly the same as one another), no further differentiation among Otto engines would be considered at Level n+3.

Certain applications seek to compare technology embodied in specific commercially available devices with all possible embodiments of those and other technologies. From a commercial perspective, this enables an organization to understand the competitive landscape in which its products and those of its existing competitors exist. Within the Knowledge Center, in the hierarchy of devices, specific commercially available embodiments exist as children beneath their generic types.

2. *Load characterization*

Load characterization describes application performance requirements. Loads can take the form of a single power level that must be achieved, or a power profile that must be delivered over time. For example, a rotorcraft power load could be expressed as a profile of shaft power level vs. flight state (e.g., hover, level-flight, climb, descent).

3. *Filters*

Filters describe inherent device constraints and application constraints. Each type of device is capable of accepting a limited set of energy types as input and is capable of providing a limited set of energy types as output. Application constraints take many forms so filters may represent a wide variety of issues. For example, the design space may include a variety of possible fuels; however, an application constraint may be that at most one logistical fuel can actually be used. Other application-constraint filters may describe the number of instances of devices embodying specific technologies (e.g., at most two engines), may describe which technologies are compatible and thus may be used within the same system, or may specify that at most one device may provide each energy type to another device. A sufficiently comprehensive and flexible set of configurable filter primitives are needed to enable expression of all relevant constraints in a simple and direct manner.

4. *Application Value Metrics*

Value metrics are the parameters of an architecture that drive its value (e.g., weight, volume, installed cost, operating cost, environmental impact, risk). These metrics characterize the most important drivers of value in an energy system. If their relative importance is known quantitatively, it can be used to form a single objective function for the energy system.

B. Architectural Enumeration

Architectural Enumeration consists of algorithms that consider all possible architectures in the design space and filter out those that are infeasible because they violate requirements or restrictions. Architectures are represented as combinations of interconnected devices that embody energy-system technologies, such as engines, generators, motors, storage devices (e.g., batteries), and waste energy recovery devices (e.g., bottoming cycles). This modularity simplifies description of new technologies. The enumeration algorithm enables consideration of all possible combinations of devices and ways to interconnect them to create an energy system. This may suggest innovative and non-intuitive architectures that have not yet been contemplated.

Architectural Enumeration uses filters to rapidly and efficiently pare down the design space by excluding architectures that violate network-integrity and application constraints. Network-integrity constraints ensure the logical connectivity of an energy system. As specified in the Knowledge Center, each device is capable of accepting a limited set of input energy types (e.g., JP-8 or diesel fuel, electric power, shaft power), and likewise, is capable of providing a limited set of output energy types. Network-integrity filters reduce the design space by enforcing these input and output constraints by ensuring that if an output of device A is connected to an input of device B, then that connection has a single energy type which is both an acceptable output of device A and an acceptable input of device B (e.g., engine shaft power output cannot be supplied as input to a chemical battery). Other qualitative network integrity filters include the following: ensuring that all available system inputs are used as an input of at least one device, ensuring that all desired system outputs are provided by at least one device, and ensuring that no subsets of the network are isolated “islands” operating without an energy source. Application constraints are specified in the Knowledge Center and applied during Architectural Enumeration (e.g., at most, one logistical fuel).

There are two possible implementations of filters, evaluative and generative. Evaluative filters require that an architecture be generated, and then proceed to evaluate its feasibility according to the application constraints. Generative filters simply avoid generation of infeasible architectures. Consequently, generative filters are computationally far more efficient than evaluative filters. However, implementation of evaluative filters is straightforward, while implementation of generative filters requires understanding of the problem structure, so that it can be used to avoid generating infeasible architectures. Computational speed is maximized by using generative filters. Figure 5 shows the decrease in computational time that has been achieved via generative filtering in an

HVAC/CHP conceptual design application (e.g., a design space with 10^{74} architectures reduced to a feasible set size of 278 architectures).

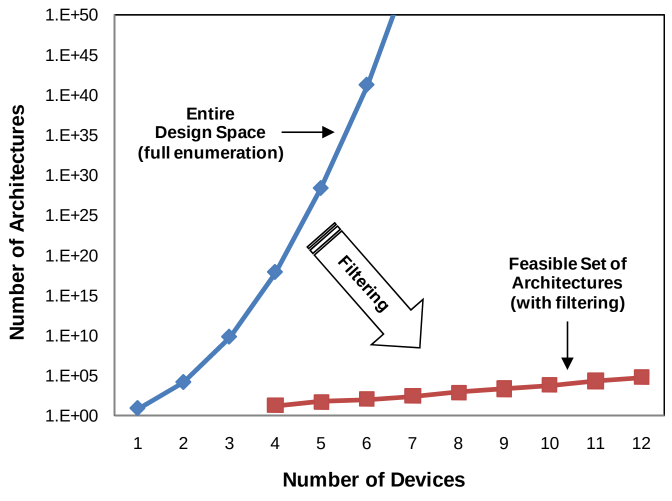


Figure 5: Effect of Filtering on Reducing Feasible Set Size

The output of Architectural Enumeration is a set of all feasible architectures. This set is typically many orders of magnitude smaller than the design space. This sparseness is what enables generative filters to be so effective at reducing computational time for Architectural Enumeration.

C. Architectural Evaluation

All feasible architectures that are identified by Architectural Enumeration are evaluated on selected metrics (e.g., complexity, efficiency, weight, operating cost, and technology readiness) using low fidelity device models to ensure speed. However, it is worthwhile to distinguish between “apparent feasibility” which is what filtering assesses, and “actual feasibility” which can only be assessed by a combination of qualitative filtering and quantitative evaluation. For example, an application constraint of total system weight <500 lbm cannot be assessed qualitatively. Assessing this constraint requires a system energy balance to be evaluated, resulting in knowledge of the power flowing along each device interconnection, from which the total power into each device can be computed, from which the devices can be sized, from which their power densities and/or energy densities can be used to compute their individual weights, from which the total system weight can be computed and compared with the 500 lbm constraint.

The most promising architectures are identified, based on the value metrics (e.g., system complexity, weight, volume, environmental impact). If the application’s value metrics can be combined quantitatively into a single objective function, then this can be used to rank candidate architectures. If there is no intuition regarding the relative importance of value metrics, then their Pareto frontier can be explored to understand the which design choices are the most important drivers of value.

Uncertainty analysis could be applied to determine the robustness of promising architectures to variations in device attributes, application constraints, ambient conditions and relative value-metric weights. Architectures could be compared based not only on their best possible system performance, but also on their average system performance over an expected range of uncertain device performance, conditions and uncertain relative importance of value metrics. Sensitivity analysis could be applied to determine which design parameters have a strong influence on the architecture’s value. The results of sensitivity analysis can be used to guide efforts to better characterize those parameters that have the greatest influence. Analysis management could avoid the need to re-run prior analyses by storing the results and conclusions of analyses and making them easily accessible. This trade of storage capacity vs. computational capacity depends on their relative availability and cost.

An interface between successive levels within Technology Screening would ease the transition of information, enabling more promising architectures to be evaluated with higher fidelity. At the interface between each level and the next layer, this involves automatically generating higher-fidelity system models, based on the devices included, their interconnection, higher-fidelity device models stored in the Knowledge Center, and initial conditions estimated from the lower-fidelity evaluation results (e.g., power levels of devices, and their sizing). A user interface would enable an analyst to specify properties that were abstracted out of the lower-fidelity models, but which are precisely the higher-fidelity elements that must be added to create a higher-fidelity system model.

D. Architectural Visualization

A graphical display enables rapid visualization of device configuration, interconnection, and system inputs and outputs as illustrated in Figure 1. The graphical user interface is helpful to understand the evaluation results and interpret the advantages and disadvantages of particular technologies and architectural features. If there is insufficient understanding of the application to create a single objective function, then the application can be considered as a multi-objective problem, and a Pareto frontier can be visualized to enable interpretation of the evaluation results so that intuition can be developed about why particular architectures are valuable.

IV. Example Application

The example application is drawn from aircraft design and focuses on power and thermal management. The system is described in terms of a set of subsystems and their interconnections via energy, fuel and data flows, including flows to and from the rest of the aircraft.

A. Architectural Enumeration

There are seven subsystems, which are shown in Table 1:

Table 1- Subsystems, their functionality and complexity

Subsystem	Function	Complexity
Engine	the focus here is on power for the other subsystems (not thrust)	20
Fuel System	meters fuel	8
Electric Power System	provides electric power for subsystems and aircraft hotel loads	10
Environmental Control System	provides aircraft cooling	10
Auxiliary Power Unit	provides auxiliary power when the Engines are not available	15
Ram Cooler	provides a heat sink	4
Actuation	provides flight control	8

There are 15 types of energy, fuel and data flows (listed in Table 2 in order of decreasing complexity):

Table 2- Inputs: Energy, fuel and data flows

Energy, fuel & data flows					Complexity			
Name	Power	Available System Inputs	Desired Output	Link color	Total	Send	Convey	Receive
Electric power	1	0	1	▨	4	50%	0%	50%
Pneumatic power	1	0	0	🟩	1	50%	0%	50%
Shaft power	1	0	0	🟧	5	33%	33%	33%
Heat	0	0	0	🟥	2	33%	33%	33%
Fuel	0	1	0	🟦	3	50%	50%	0%
Metered fuel	0	0	0	🟦	3	50%	50%	0%
Ram air	0	1	0	🟦	1	50%	0%	50%
Conditioned ram air	0	0	0	🟦	1	50%	0%	50%
Cooling	0	0	1	🟦	1	75%	25%	0%
Flight control	0	0	1	🟪	0	100%	0%	0%
Start pneu power	1	0	0	🟩	1	50%	0%	50%
Start shaft power	1	0	0	🟧	5	33%	33%	33%
Thrust	1	0	0	🟫	0	100%	0%	0%
Aux shaft power	1	0	0	🟧	5	33%	33%	33%
Aux pneu power	1	0	0	🟩	1	50%	0%	50%

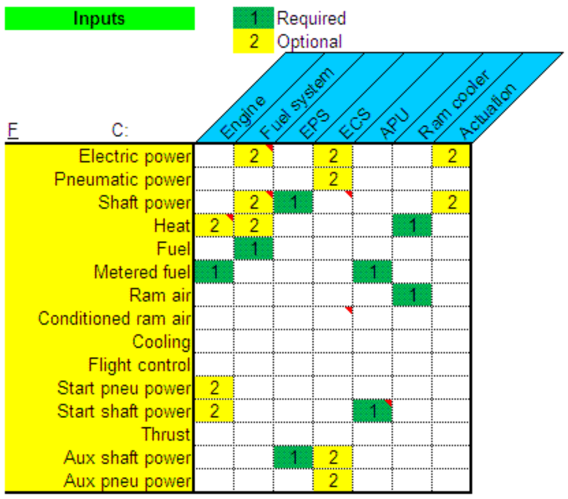
Each flow has a name, may be a form of power, may be an available system input or a desired system output, has a link color (for display on the architectural layout), has a total complexity (according to the Link Types complexity principle), and has a breakdown of that total complexity into the complexity to send, convey and receive that type of flow.

The system requirements and constraints are:

- Available System Inputs:
  - Fuel
  - Ram Air
- Required System Outputs:
  - Electric Power
  - Cooling
  - Flight Control
- Required Flows:
  - Fuel System –(Metered Fuel)→ Engine
  - Fuel System –(Metered Fuel)→ APU
  - EPS –(Start Shaft Power)→ APU
  - APU –(Aux Shaft Power)→ EPS
- Prohibited Flows:
  - Engine –(Heat)→ ECS
  - Engine –(Heat)→ Ram Cooler
  - Fuel System –(Heat)→ Engine
  - EPS –(Heat)→ Engine
  - EPS –(Heat)→ ECS
  - EPS –(Heat)→ Ram Cooler
  - Actuation –(Heat)→ Engine
- Each flow has a name, may be a form of power, may be an available system input or a desired system output, has a link color (for display on the architectural layout), has a total complexity (according to Link types complexity principle), and has a breakdown of that total complexity into the complexity to send, convey and receive that type of flow.

Each type of subsystem can accept certain types of energy, fuel and data flows (listed in Table 3):

Table 3 - Subsystem flow inputs



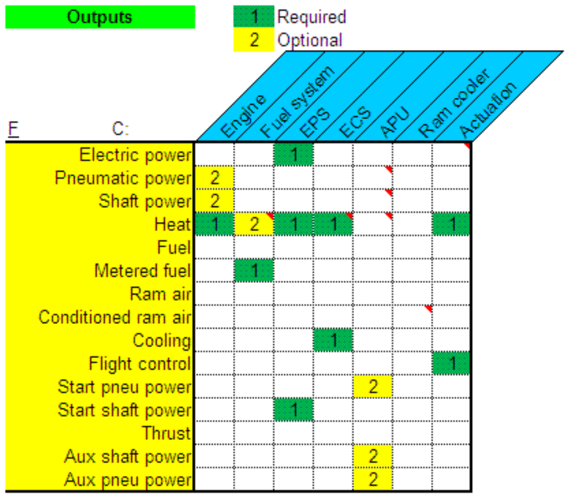
Inputs

1	Required
2	Optional

F C:	Engine	Fuel system	EPS	ECS	APU	Ram cooler	Actuation
Electric power		2		2			2
Pneumatic power				2			
Shaft power		2	1				2
Heat	2	2				1	
Fuel		1					
Metered fuel	1				1		
Ram air						1	
Conditioned ram air							
Cooling							
Flight control							
Start pneu power	2						
Start shaft power	2				1		
Thrust							
Aux shaft power			1	2			
Aux pneu power				2			

Likewise, each type of subsystem can produce certain types of energy, fuel and data flows (listed in Table 4):

Table 4 - Subsystem flow outputs



Outputs

🟩 1	Required
🟨 2	Optional

F C:	Engine	Fuel system	EPS	ECS	APU	Ram cooler	Actuation
Electric power			1				
Pneumatic power	2						
Shaft power	2						
Heat	1	2	1	1			1
Fuel							
Metered fuel		1					
Ram air							
Conditioned ram air							
Cooling				1			
Flight control							1
Start pneu power					2		
Start shaft power			1				
Thrust							
Aux shaft power					2		
Aux pneu power					2		

The architectural enumeration for this problem produced ~20,000 feasible architectures in less than 1 minute. These architectures pass all of the specified requirements and constraints, so they are feasible in the sense that they are configured feasibly. At this point in the process, neither complexity nor other value metrics such as performance, weight, durability, etc. have been evaluated.

Figure 6 illustrates one of the feasible architectures. It will be examined further below.

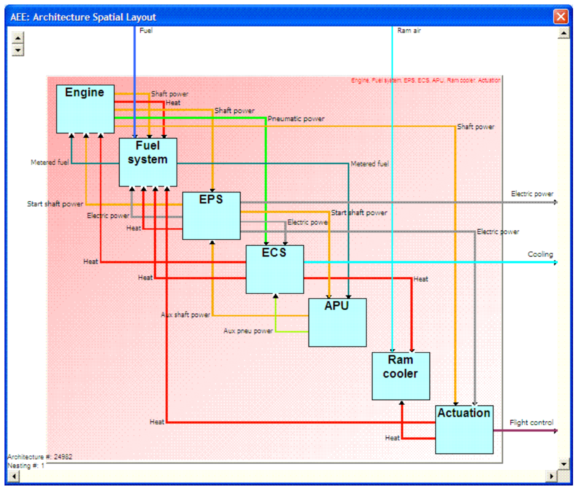
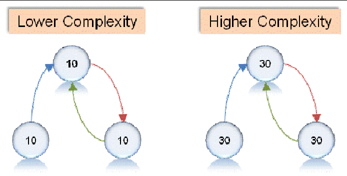
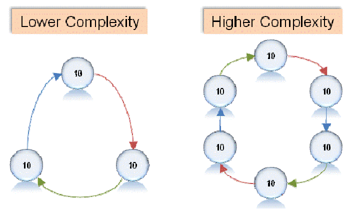
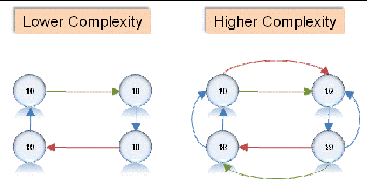
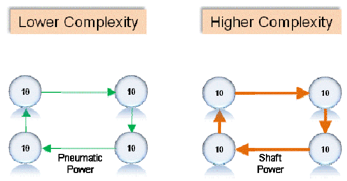


Figure 6 - A feasible architecture

B. Complexity Metric

A complexity metric can be defined based on a set of complexity principles, such as those shown in Table 5.

Table 5 - Complexity Principles

Principle	Description	Diagram
Node complexity	A system's complexity increases as the complexity of its constituent parts increases.	 Lower Complexity Higher Complexity 10 10 10 30 30 30
Number of nodes	The complexity increases as the number of constituent parts increases.	
Number of links	The complexity increases as the number of interconnections increases.	
Types of links	The complexity that each type of interconnection adds to a system is based on the type of energy/fuel/data flow.	 Lower Complexity Higher Complexity Pneumatic Power Shaft Power

A complexity metric defined in this manner can be applied to an architecture, such as the one shown in Figure 6, by combining the complexity introduced according to each complexity principle, with the result shown in Figure 7. Note that in Figure 7 the complexity metric values for each subsystem are shown in the lower right corner of each subsystem box, and that the overall complexity metric value for the system architecture is shown in the lower right corner of the overall system box.

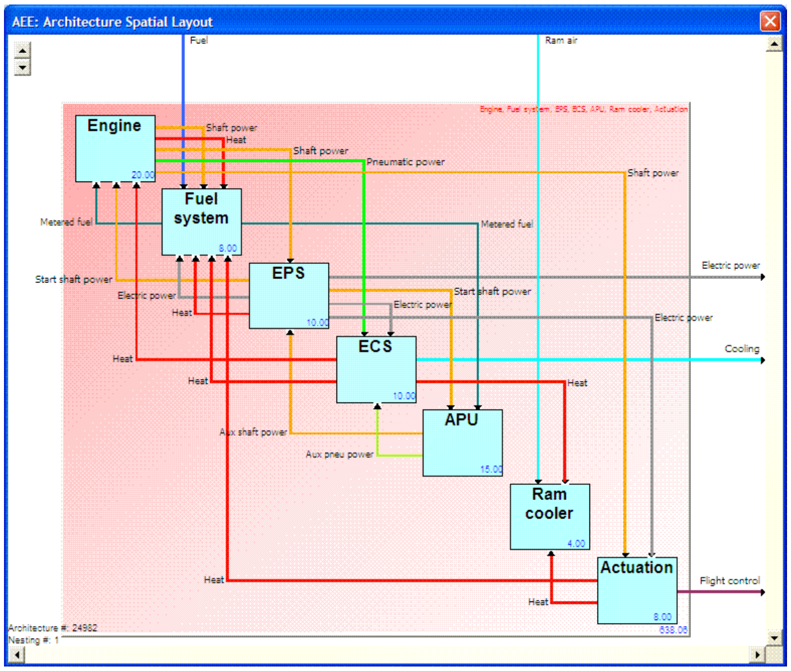


Figure 7 - Complexity metric applied to a feasible architecture

As described in [15], an architecture can be hierarchically clustered, using spectral graph partitioning methods coupled with complexity principles. Figure 8 shows a clustered version of the architecture shown in Figure 6 with a significantly lower complexity.

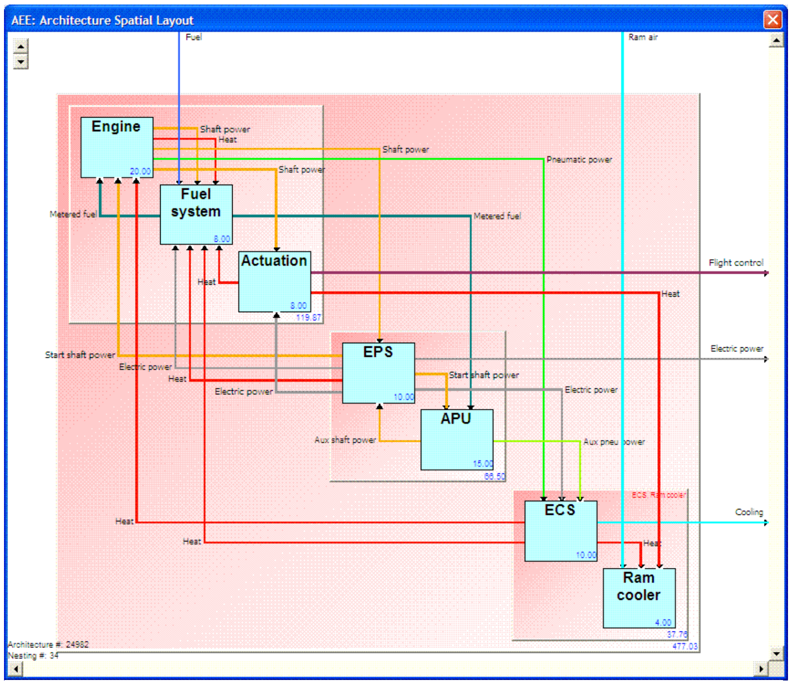


Figure 8 - Aerospace power & thermal management architecture clustered using BestWCut result.

V. Conclusion

A method for enumerating the sparse feasible subset of very large architectural design spaces has been described. This method is intended to be used at the start of architecting large systems so that all feasible architectures are considered and the best low-complexity (low cost) architecture can be found. This method for finding feasible architectures differs from prior methods in that it is comprehensive, considering the entire design space, rather than searching through the design space, exploring only a subset of the possible architectures; AEE ensures that all feasible architectures are found in a timely fashion.

Acknowledgments

The development of the methods and results contained in this study were sponsored under the DARPA contract: "Abstraction Based Complexity Management" #FA9550-07-C-0024. [8]

References

[1]Zeidner, L.E., St. Rock, B.E., Desai, N.A., Reeve, H.M., and Strauss, M.P., "Application of a Technology Screening Methodology for Rotorcraft Alternative Power Systems," *AIAA Aerospace Sciences Meeting*, AIAA-2010-1505, Orlando, FL 2010.

[2]Zeidner, L.E., Becz, S.B., Reeve, H.M., and Khire, R., "Design Issues for a Bottom-Up Complexity Metric Applied to Hierarchical Systems," 10th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference, 2010

[3]Zeidner, L.E., Banaszuk, A., and Becz, S.B., "System Complexity Reduction via Spectral Graph Partitioning to Identify Hierarchical Modular Clusters," 10th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference, 2010.

[4]Kerzhner, A. A., Paredis, C. J., "Using Domain Specific Languages to Capture Design Synthesis Knowledge for Model-Based Systems Engineering", Proceedings of the ASME 2009 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference (IDETC/CIE), 2009 August 30 - September 2, 2009, San Diego, CA.

[5]Sangiovanni-Vincentelli, A., "Defining Platform-based Design", EEDesign of EETimes, February 2002. [Online] [Cited: October 12, 2009] http://www.gigascale.org/pubs/141/platformv7eetimes.pdf

[6]Carloni, L. P., De Bernardinis, F., Pinello, C., Sangiovanni-Vincentelli, A. L., and Sgroi, M., "Platform-Based Design for Embedded Systems", http://www1.cs.columbia.edu/~luca/research/pbdes.pdf.

[7]Arena, M. V., Younossi, O., Brancato, K., Blickstein, I., Grammich, C. A., "Why Has the Cost of Fixed-Wing Aircraft Risen?" RAND Corporation, 2008, http://www.rc.rand.org/pubs/monographs/2008/RAND_MG696.pdf.

[8]Becz, S., Pinto, A., Zeidner, L.E., Khire, R., Reeve, H.M., "Design System for Managing Complexity in Aerospace Systems", 10th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference, 2010.

F.6 Design Issues for a Bottom-Up Complexity Metric Applied to Hierarchical Systems

Design Issues for a Bottom-Up Complexity Metric Applied to Hierarchical Systems

Lawrence E. Zeidner[1], Sandor B. Becz,[1] Hayden M. Reeve[1], and Ritesh Khire[1]
United Technologies Research Center, East Hartford, CT 06108, USA

The cost of large complex systems can be reduced by using system complexity as a cost proxy during the initial stages of system architecting. This requires evaluation of system complexity in terms of information available during those early stages. This paper presents the design of complexity metrics in terms of the aspects of systems that drive system complexity. System modularity is a well-known technique for reducing system complexity by clustering system elements into hierarchical modules. This paper also explains how complexity metrics can be designed to be applicable for hierarchical systems.

Nomenclature

AFRL	=	Air Force Research Lab
INVENT	=	Integrated Vehicle Energy Technology
CCI	=	Capability/cost index
N_t	=	total number of nodes
L_t	=	total number of links

I. Introduction

Significant technological and architectural changes have been introduced into aerospace systems, over the past several decades, in an effort to improve the performance and capability of new platforms. These changes have led to an exponential growth in the complexity of modern aerospace platforms and accompanying design and development challenges. These increases in technical complexity have been accompanied by increases in the complexity of system requirements and organizational partnerships. Complexity in requirements stems from the need to meet multiple present and future mission requirements. Analysis by the RAND Corporation found that the largest component in the growth in the cost of fixed-wing aircraft has come from increased complexity[1]. As systems have become more complex their development cost has increased and the predictability of their development cost has decreased. Managing and minimizing complexity of new system development offers the ability to reduce the magnitude and increase the predictability of development cost. Fundamental architectural decisions made early in the design process have a major impact on system complexity; however, system architects lack quantitative feedback on the complexity of their architectures.

During the early design stages of large systems, systems are architected in terms of functional subsystems (e.g., Electrical Power System, Environmental Control System, Auxiliary Power Unit) and their interconnections as shown in Figure 1. This standard top-down design approach is a divide-and-conquer strategy that defines the sub-system boundaries as an interface to decouple the overall design problem into two sequential sub-problems. The first sub-problem is to design a system of functional subsystems based on overall system requirements, resulting in a set of requirements for each subsystem. The second sub-problem is to design each functional subsystem in terms of its required functionality. This approach has also enabled airframers to focus on the large-scale system-integration problem, while outsourcing subsystem design to subcontractors, with the subsystem requirements serving as the organizational interface. Since these two sub-problems are not really decoupled, subsystem interactions necessitate design iteration to meet system-level requirements. Because functional subsystems are designed separately, these iterations often occur late in the design process and are a significant cause of schedule and cost overruns.

One approach to design an overall system comprised of functional subsystems (the first sub-problem) is to enumerate and evaluate all feasible solutions to this problem, as described in [1]. This approach has typically been

[1] Research Engineer, Thermal Management, United Technologies Research Center, 411 Silver Lane, MS 129-89, East Hartford, CT, 06108.

avoided due to the exponential size of the design space. However, it has recently been shown that the feasible set is actually quite sparse, many orders of magnitude smaller than the overall design space, and that generative filters can be used to identify the entire set of feasible solutions directly, rather than searching through the design space for feasible solutions. [1] Using this approach, each architecture in the feasible set is specified in terms of the type of information depicted in Figure 1.

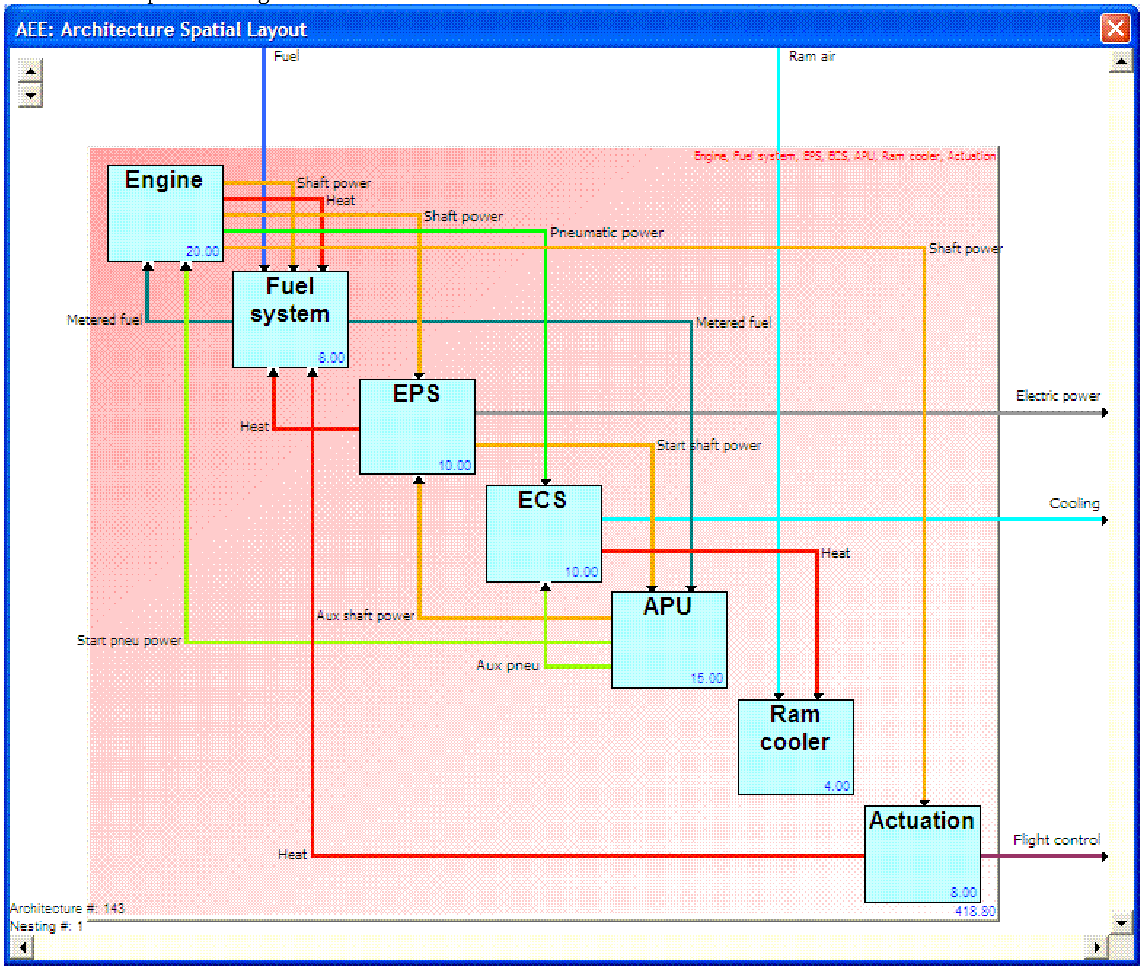


Figure 1 - An aerospace system architecture specified in terms of its functional subsystems and their interconnections, with a complexity metric assessed for each subsystem and for the overall system.

The complexity of each architecture depends not only upon its constituent technologies and its interconnections, but also on how and to what degree the architecture is organized hierarchically into modules. Each architecture can be decomposed into many different hierarchical configurations of modules, each with its own degree of complexity. This paper presents the design issues pertaining to the complexity metrics suitable for hierarchical system configurations. These metrics roll up the complexity within each hierarchical module based on the complexity of its internal elements and interconnections.

The ability to quantify and thereby minimize complexity of advanced systems throughout the design process is required. This, in turn, requires both abstract (i.e. low fidelity) complexity metrics as a leading indicator of complexity for use early in the process during configuration selection, cost estimation, and bidding and proposing, all the way to detailed complexity metrics for use during detailed design. This paper will focus on the design of abstract complexity metrics used early in system architecting.

When we make decisions involving system architectural alternatives, it is important to evaluate how difficult the resulting product-development task may be and how long it may take. This evaluation is crucial in allocating resources and in determining the appropriate cost and timing of product development. System complexity metrics have a long history in the context of software quality control [7]. Typical metrics - such as McCabe Cyclomatic Complexity Metric [4, 6] of system complexity, use the graph structure of the underlying system in order to formulate a metric. There are a number of different studies some of them in large industrial concerns such as IBM and Hughes Aircraft [2] that attempt to correlate the metric with the number of errors (and thus decreased productivity) in the code, with mixed results. Shepperd and Ince analyzed the three most widely used software complexity metrics and identified that each was poorly formulated, involved implicit assumptions that were not characteristic of the ways in which the metrics were being applied, and had misleading or faulty statistical validation. [6]

There has been considerable work defining complexity[14] but less effort on constructing a quantitative metric that can guide design decisions. Kim and Wilemon [5] provide a review of complexity definitions that have been developed. These definitions cover the range of project, product, R&D/innovation, integration, and market areas and the definitions include the following core elements:

- *Numbers:* Number of different disciplines or departments involved. Number of parts, technologies, or functions required in a product.
- *Degree of Interdependency:* Level of interdependency among the domains, functions, or disciplines involved.
- *Intricacy or difficulty:* Novelty of project (minor modifications and growth and derivative versions versus clean sheet designs with untested technologies)
- *Limitations:* A compounding factor that can increase the complexity in the areas above. Examples include: limited time to market, tight performance requirements (weight, thrust), stringent constraints (thermodynamic limitations).

Other definitions have been used in the aerospace field. AFRL's INVENT program[8] views complexity as equivalent to the inflexibility of a design to meet future growth requirements. That is, how tightly integrated the design space is with respect to change. Arena et al.[10] used the term complexity loosely to refer to the increased capability of aircraft. Jones et al. [9] developed a quantitative metric to estimate the cost of large scale systems by developing a metric based on the number of nodes and links within a system. To effectively manage complexity in the future domain specific standard measures of complexity are needed that would allow competing offerings to be ranked, similar to the cost capability index (CCI) used to quantify the capability of propulsion systems. More importantly, identifying the key attributes and contributing factors that create or amplify complexity (and therefore development cost and risk) is a key requirement to being able to manage and minimize complexity.

Prior art in the design of complexity metrics includes metrics for:

- software quality control,
- complex task environments, notably air-traffic control: Histon defined structure-based abstractions about the complex environment that can be used to improve air-traffic controller's situational awareness so that they can plan and project future conditions. [11] These abstractions are akin to our complexity drivers in that they form a language with which to describe high-level features that drive the level of complexity. They both form the basis for quantitative analysis and combination into a single metric, and form the kernel of a complexity-simplification approach, if they're new abstractions to those performing the tasks.
- Supply chains: Abbasi [12] describes these complex socio-technical, chaotic, self-organizing, non-deterministic systems based on properties described in Pavard.

The remaining sections of this paper are organized as follows. Sections II and III will define the problem and present a solution approach. Section IV will provide an example application of this solution approach. The paper concludes with a summary and with suggested extensions.

II. Problem Definition

The technical challenge is to formulate a system complexity metric that serves as a good early proxy for total system cost, so that total system cost can be balanced against other value metrics for early architecture decisions. System architects have ways to predict performance, weight, volume and other value metrics, but complexity as a proxy for total system cost as well as development cost and development time are needed.

Early in the design of an overall system comprised of functional subsystems, the subsystems themselves have not yet been designed or sized. The information available about each subsystem consists of the sub-system technology and its energy, fuel and/or data input and output flows. The information available about the interactions among the sub-systems, and between the subsystems and the system's environment, consists of the type of energy, fuel or data flowing through each interconnection. This information, together with the modular system hierarchy is the information available to a complexity metric.

III. Solution Approach

Shepperd and Ince [6] describe the five specific components of a well-formed model:

1. Purpose: the purpose of the model must be known to determine which factors must be included and which can be excluded.
2. Theory & Relationships: the model embodies a theory, an idea, expressed in the form of relationships established between inputs and outputs, and may involve parameters.
3. Measurement & Prediction: the measurement process describes how reality maps into the model's inputs, and the prediction process describes how the model's outputs map back into reality.
4. Limitations: explicit assumptions regarding applicability of the relationships to specific domains.
5. Veracity: the confidence that one can have that the model will accurately predict real-world behavior.

The solution approach for designing a bottom-up complexity metric for hierarchical systems will be described in terms of these five components in the following sections.

A. Purpose

The purpose of a complexity metric, used early in system architecting, is to provide guidance to support design decisions that will select lower-complexity architectures, to result in lower system development time and cost and lower total system cost.

The context in which this complexity metric is intended to be used is early in the design process, when basic system architectural alternatives are being considered. The complexity metric is one of three techniques that are employed together to reduce the enormous design space to a manageable size before proceeding to a lower level of abstraction, using higher fidelity models:

- Architectural enumeration [1]: The entire architectural design space is considered rapidly to identify the feasible set of architectures that meet configuration requirements. The architectural design space is enormous, but the feasible set is extremely sparse. Generative filters are used to rapidly consider the entire design space and identify the entire feasible set. The architectural design space is specified by identifying the technology alternatives at the relevant level (e.g., device, subsystem), the energy/fuel/data flow types between devices or subsystems, and the energy/fuel/data inputs and outputs of each technology alternative. Architectural enumeration results in a list of feasible architectures.
- Spectral graph partitioning [Spectral]: Spectral methods are applied iteratively to each architecture to cut the architecture into hierarchical modular clusters that minimize the complexity of that architecture as measured by the complexity metric. The partitioning method enables principles of cost due to complexity to be embodied in an adjacency matrix used as input. The adjacency matrix describes the energy/fuel/data flows between devices or subsystems. Principles involving flows that cut across multiple levels in the hierarchy are embodied in the way those flows are quantified as the spectral methods are applied iteratively to create the hierarchy.
- Complexity metric: A complexity metric embodies principles of cost due to complexity. This is the subject of this paper.

B. Theory & Relationships

There are multiple principles that describe how elements of complexity drive system cost. These principles are combined to form the theory for a complexity metric. The following are examples of principles believed to be relevant to aerospace systems; however, the point of this paper is not the specific principles themselves, but how such principles can be combined to create a complexity metric that embodies them.

- Node complexity: A simple principle describing the way in which complexity propagates through a system is that a system's complexity increases as the complexity of its constituent parts increases. For example, a system of three interconnected devices, each of complexity 10 has lower complexity than an otherwise identical system of three interconnected devices, each of complexity 30, as shown in Figure 2.

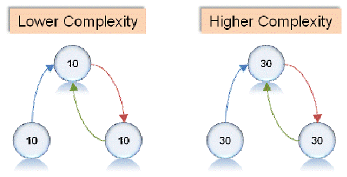


Figure 2 - Node complexity

- Number of nodes: Another simple principle describing complexity propagation is that, at any hierarchical level within a system, the complexity increases as the number of constituent parts increases. Applying this principle at the device level, a subsystem consisting of three interconnected devices of complexity 10 has lower complexity than an otherwise identical subsystem consisting of six interconnected devices of complexity 10, as shown in Figure 3.

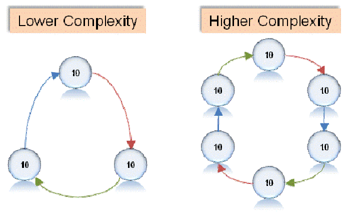


Figure 3 - Number of nodes

- Number of links: At any hierarchical level within a system, the complexity increases as the number of interconnections increases. Applying this principle at the system level, a system of three nodes interconnected by 2 links has lower complexity than an otherwise identical system of three nodes interconnected by 6 links, as shown in Figure 4.

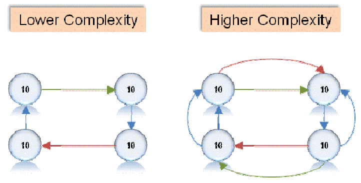


Figure 4 - Number of links

- Types of links: Within an architectural description, interconnections are used to represent flows of energy, fuel, or data between devices or subsystems. Implementing these different types of flows requires different methods, each with its own level of complexity and associated cost. Consequently the complexity that each type of interconnection adds to a system can be determined based on the type of energy/fuel/data flow. For example, flows of pneumatic power require relatively simple air pipes from the interconnection source to the target, while flows of shaft power require strong metal shafts from source to target, along with various types of bearings along the path, and often gearboxes to achieve appropriate rotational speed for transmission, as shown in Figure 5.

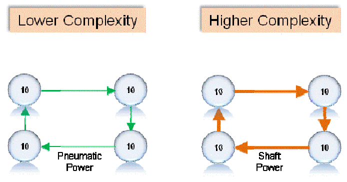


Figure 5 - Types of links

- Roll-up: Within the hierarchical modular clustering of an architecture, a module encapsulates the complexity of the elements inside it and their interconnections with one another and with the module's environment. Complexity of a module is based on various principles in combination. At a given level within the hierarchy, a module's complexity acts as an interface between two levels. At the lower level, the module's complexity is determined as an output. At the next-highest level, the module's complexity is assumed as an input, regardless of how its configuration led to its complexity, as shown in Figure 6.

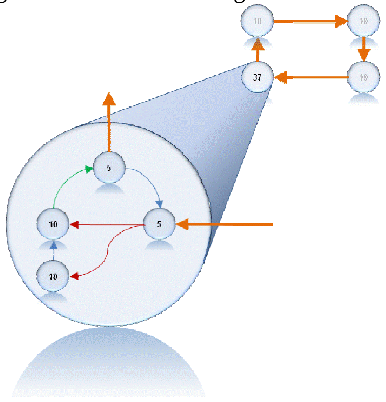


Figure 6 - Roll-up

- Shared conveyance: In hierarchical modular systems, multiple interconnections of the same energy/fuel/data flow type may link two embedded modules, as shown in Figure 7. For example, at the system level, shaft power may link an engine to another embedded module, such as a combined Environmental Control System and Auxiliary Power Unit. It may be possible to combine the two flows at the source, by means of a gearbox, convey one shared shaft-power link from source to target, and then decompose that combined shaft power into two separate flows for the ECS and APU, by means of another gearbox. For links travelling relatively long distances on an aircraft from source to target, this shared conveyance may reduce cost significantly and present other benefits such as weight reduction, while for very short distances, it may not be worth the cost of the gearboxes. For data flows, multiplexing is an example of shared conveyance of multiple data channels along one physical pathway, with associated mux/demux costs at the source/target.

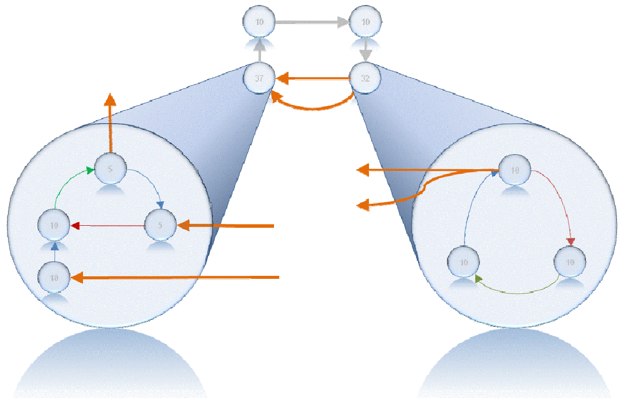


Figure 7 - Shared conveyance

- Pass-through: In hierarchical modular systems, an interconnection may pass from one embedded subsystem to another as shown in Figure 8. At the hierarchical level in which both subsystems are embedded and interconnected, little or no value-added work is done to the flow, so little or no cost is added. These flows are simply passing through from a lower level, and to a lower level, surfacing only to link across the system. Consequently, at this level, the interconnection link adds relatively little complexity to the system.

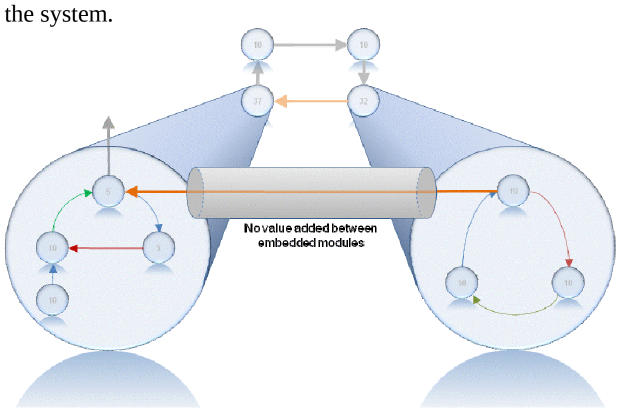


Figure 8 - Pass-through

- Complexity isolation: Clustering nodes and their associated interconnections into hierarchical modular clusters has the advantage of localizing complexity so that other modules and higher levels may have lower complexity as shown in Figure 9.

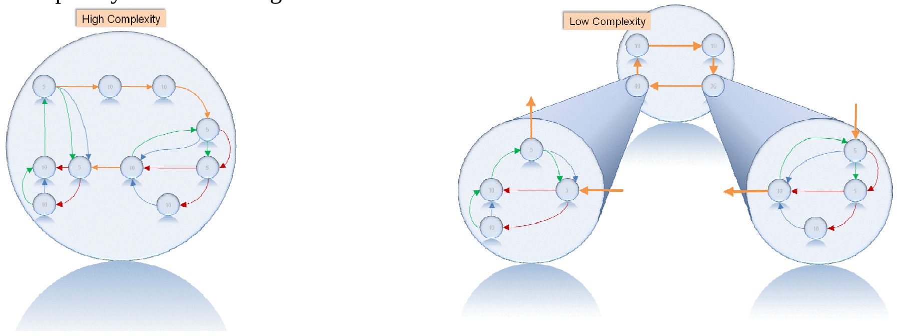


Figure 9 - Complexity isolation

- Clustering cost: While complexity isolation due to clustering is beneficial, each module defines an interface, requires design, test, and sustainment, along with associated costs, and imposes constraints that limit flexibility later in the product life cycle. So while clustering is beneficial, it must be balanced in the complexity metric by an associated cost, so that an optimum degree of clustering will exist for a given architecture.

Complexity principles can be embedded in a complexity metric. For example, Jones, et al. [9] present a complexity metric that embodies a few of these principles: node complexity, number of nodes, and number of links.

Their complexity metric is:

$$RDT\&E\$ = aN_e^b \qquad \text{where}$$

$$N_e = (aN_{s/r} + bN_s + cN_r)\left(\frac{L_t/N_t}{\text{avg}\,(L_t/N_t)}\right)^d$$

Capture complexity associated with type of Nodes

Capture the connectivity complexity of the system

The node complexity principle is embodied in the distinction between three types of nodes: receivers (N_r), transmitters (N_s), and transceivers ($N_{s/r}$), and their multiplication by three separate complexity factors, h, g and d. The number of nodes principle is embodied in multiplication by the term ($dN_{s/r} + gN_s + hN_r$). The number of links principle is embodied by the term L_t.

The metric presented by Jones does not embody the types of links principle. However, we can adapt it to embody this principle by changing the interpretation of L_t to represent the sum of the complexity of each link, based on energy type. For example, we can assume that the relative complexity of the various types of links is:

5: Shaft power
4: Electric power
3: Fuel
2: Heat
1: Pneumatic power

Figure 10 shows two systems that have significantly different scores on Jones' complexity metric, since one has many more links than the other. However, both of these systems have the same score on the metric adapted above to embody the types of links principle, since one had many simple links and the other had few, but complex links.

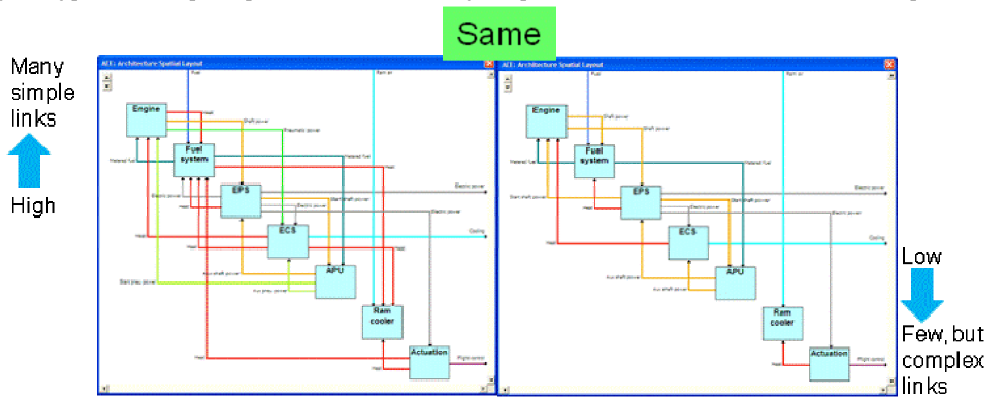


Figure 10 - Two systems that score differently on Jones metric and identically once the metric is adapted to embody the types of links principle.

Figure 11 shows two systems that have identical scores on Jones' complexity metric, since they have the same number of nodes and links. However, these two systems have significantly different scores on the metric adapted above to embody the types of links principle, since one has simple links and the other has complex links.

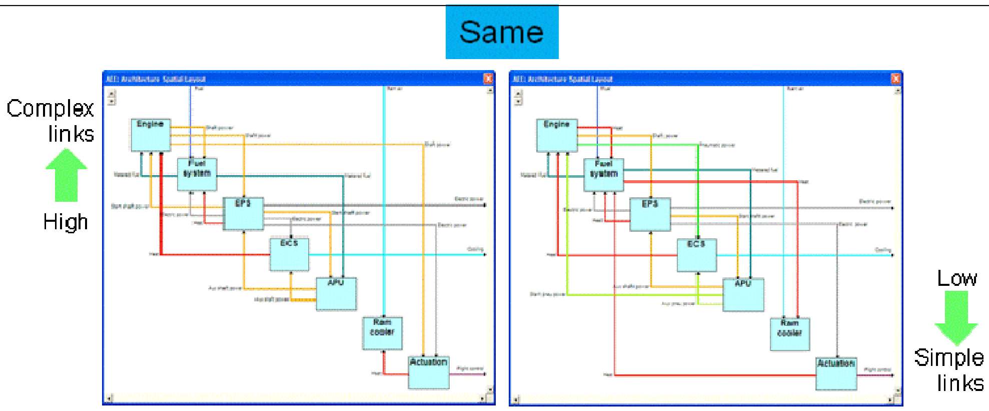


Figure 11 - Two systems that score identically on Jones metric and differently once the metric is adapted to embody the types of links principle.

A bottom-up complexity metric rolls up sub-system complexity, at each level, to the next highest level. Figure 12 shows a system without any hierarchical modular clustering. The subsystem complexities roll up to the system level and are computed along with the link weights. The complexity is noted in the bottom left of each subsystem and module.

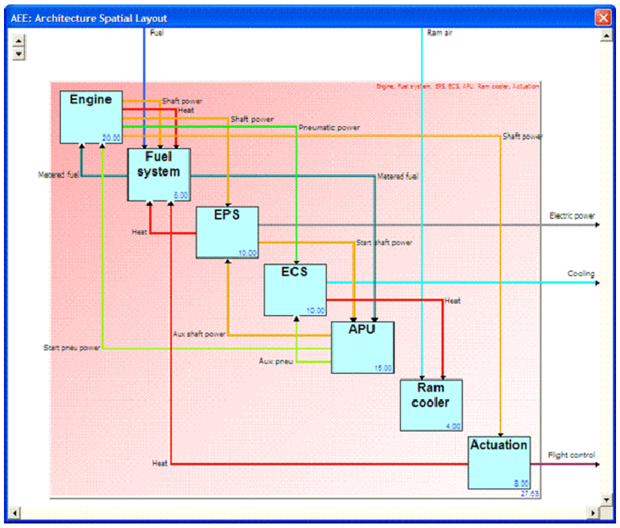


Figure 12 - System with bottom-up metric roll up but no hierarchical modular clustering.

Figure 13 shows the same system with one modular cluster and Figure 14 shows the same system with three hierarchical modular clusters.

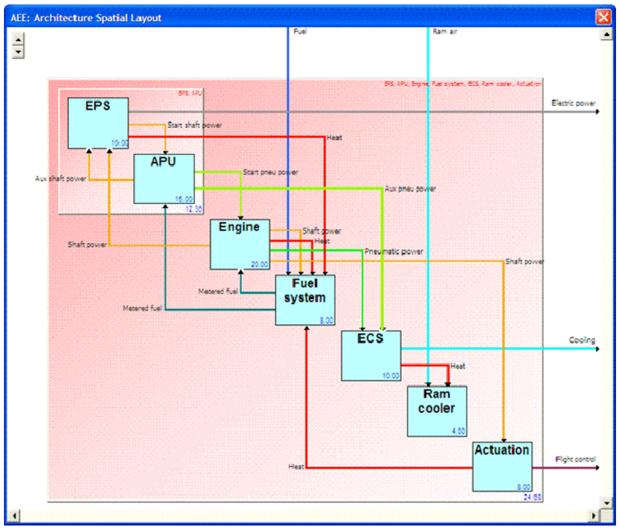


Figure 13 - System with bottom-up metric roll up and one module.

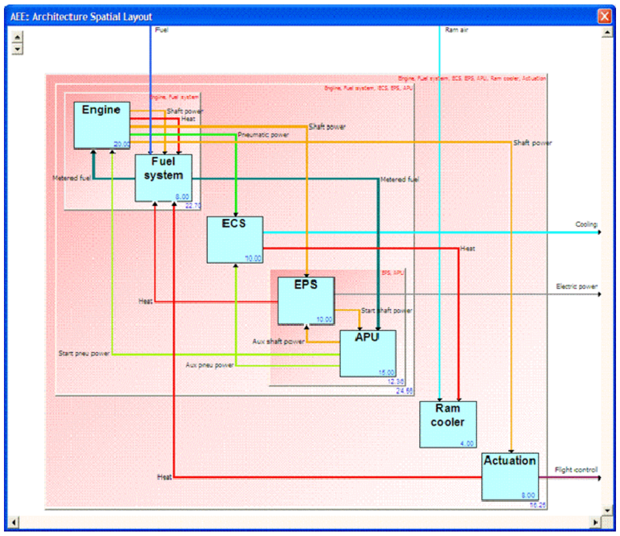


Figure 14 - System with bottom-up metric roll up and three hierarchical modules.

To summarize, a set of principles was identified that represent system features that drive complexity and ultimately system cost. These principles are representative, but not exclusive. These principles were embodied in a complexity metric that explicitly included node complexity. The complexity metric was able to be rolled up from the lowest level up to the system level due to the explicit inclusion of node complexity in its formulation.

C. Measurement & Prediction

The complexity metric has the following inputs: node complexity, link complexity, number of nodes, number of links, situation characterized as an opportunity for shared conveyance (multiple links of the same flow type between the same two embedded modules), and situation characterized as a flow through (links between two embedded modules). These can be recognized by laying out the architecture and noting the hierarchical modular clustering if any.

The computed complexity metric output is a result that can be used in different ways, depending upon its veracity (see section E below). If the metric is expressed directly in terms of cost and has been adequately validated, then its results can be used as direct predictions within suitable error bounds. The metric may be expressed in terms of complexity rather than cost, and may indicate the relative complexity between alternative architectures, without the need to have strong absolute accuracy.

D. Limitations

There are assumptions underlying the theory, relationships, measurement and prediction. These assumptions limit the applicability of the results, though these limitations may not necessarily be unduly constraining for the intended use of the metric. For example, Jones' metric explicitly assumes that all nodes belong to one of three distinct classes: receivers, transmitters and transceivers. This explicit assumption is meaningful for the class of DoD programs that Jones used to build the metric; however, the assumption does not apply to the class of aerospace power and thermal management systems, which are comprised of nodes such as engines, fuel systems, and actuation. An implicit assumption underlying the Jones' metric is that all links flow data. That assumption is another one that is not applicable to aerospace power and thermal management systems, which involve flows of fuel, heat, shaft power, pneumatic power, etc. An explicit assumption stated in the explanation of the shared conveyance principle is that the benefit of shared conveyance depends to some degree on the distance between source and target.

E. Veracity

Veracity refers to the degree of confidence that one should have in the results of the metric when it is applied within the range of limitations described above (section D). For example, Jones' metric was created statistically based upon data from a variety of DoD programs. Jones provides an error quantification along with the metric. This can be used to understand the resolution of prediction accuracy so that the metric is not used to distinguish alternatives that are more similar than this level of accuracy. The principles listed above and the resulting adapted metric have not been validated and should therefore not be used for decisionmaking until validation has been conducted. They are intended to be illustrative of how complexity principles can be embodied in a complexity metric.

IV. Example Application

The example application is drawn from aircraft design and focuses on power and thermal management. The system is described in terms of a set of subsystems and their interconnections via energy, fuel and data flows, including flows to and from the rest of the aircraft.

A. Architectural Enumeration

There are seven subsystems:

Table 1- Subsystem functions and complexity

Subsystem	Function	Complexity
Engine	the focus here is on power for the other subsystems (not thrust).	20
Fuel System	meters fuel	8
Electric Power System	provides electric power for subsystems and aircraft hotel loads	10
Environmental Control System	provides aircraft cooling	10
Auxiliary Power Unit	provides auxiliary power when the Engines are not available	15
Ram Cooler	provides a heat sink	4
Actuation	provides flight control	8

There are 15 types of energy, fuel and data flows (listed in Table 2 in order of decreasing complexity):

Table 2- Inputs: Energy, fuel and data flows

Energy, fuel & data flows					Complexity			
Name	Power	Available System Inputs	Desired Output	Link color	Total	Send	Convey	Receive
Electric power	1	0	1	▨	4	50%	0%	50%
Pneumatic power	1	0	0	🟩	1	50%	0%	50%
Shaft power	1	0	0	🟧	5	33%	33%	33%
Heat	0	0	0	🟥	2	33%	33%	33%
Fuel	0	1	0	🟦	3	50%	50%	0%
Metered fuel	0	0	0	🟦	3	50%	50%	0%
Ram air	0	1	0	🟦	1	50%	0%	50%
Conditioned ram air	0	0	0	🟦	1	50%	0%	50%
Cooling	0	0	1	🟦	1	75%	25%	0%
Flight control	0	0	1	🟪	0	100%	0%	0%
Start pneu power	1	0	0	🟩	1	50%	0%	50%
Start shaft power	1	0	0	🟧	5	33%	33%	33%
Thrust	1	0	0	🟫	0	100%	0%	0%
Aux shaft power	1	0	0	🟧	5	33%	33%	33%
Aux pneu power	1	0	0	🟩	1	50%	0%	50%

Each flow has a name, may be a form of power, may be an available system input or a desired system output, has a link color (for display on the architectural layout), has a total complexity (according to the Link Types complexity principle), and has a breakdown of that total complexity into the complexity to send, convey and receive that type of flow.

The system requirements and constraints are:

- Available System Inputs:
  - Fuel
  - Ram Air
- Required System Outputs:
  - Electric Power
  - Cooling
  - Flight Control
- Required Flows:
  - Fuel System –(Metered Fuel)→ Engine
  - Fuel System –(Metered Fuel)→ APU
  - EPS –(Start Shaft Power)→ APU
  - APU –(Aux Shaft Power)→ EPS
- Prohibited Flows:
  - Engine –(Heat)→ ECS
  - Engine –(Heat)→ Ram Cooler
  - Fuel System –(Heat)→ Engine
  - EPS –(Heat)→ Engine
  - EPS –(Heat)→ ECS
  - EPS –(Heat)→ Ram Cooler
  - Actuation –(Heat)→ Engine

The architectural enumeration for this problem produced ~20,000 feasible architectures in less than 1 minute. These architectures pass all of the specified requirements and constraints, so they are feasible in the sense that they are configured feasibly. At this point in the process, neither complexity nor other value metrics such as performance, weight, durability, etc. have been evaluated.

Figure 15 illustrates one of the feasible architectures. It will be examined further below.

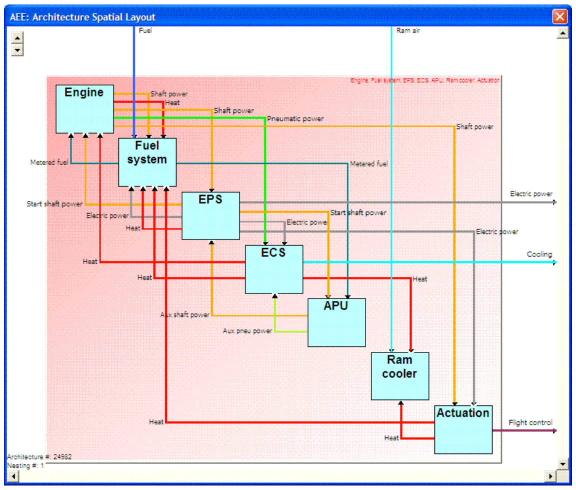


Figure 15 - A feasible architecture

B. Complexity Metric

A complexity metric is applied to an architecture, such as the one shown in Figure 15, by adding the complexity introduced according to each complexity principle, with the result shown in Figure 16. For example, using the complexity principles described in this paper, the complexity metric is computed as the following sum (Roll-up principle): **638.06** = 0.7 + 75 + 494.11 + 68.25

- Number of Nodes: A weighting factor is applied to the number of nodes. This is a relatively small contribution to the overall complexity since there are few subsystems and the node complexity principle is of greater importance for this application): **0.7** = 0.1 (weighting factor) * 7 (# of nodes)
- Node Complexity: The subsystem complexity values are added: **75** = 20 + 8 + 10 + 10 + 15 + 4
- Number of Links: A weighted sum of the number of system input links, output links, and internal links is computed, emphasizing the internal links. This weighted sum is then raised to an exponent to reflect the exponential growth of complexity with the number of links: **494.11** = ((0.5*2: weighted # of system input links) + (3*20: weighted # of internal links) + (0.5*3: weighted # of system output links))^1.5 (exponential growth with weighted # of links)
- Types of Links: The flow-type complexities are added for system inputs, internal links and system outputs: **68.25** = .5 + 64 + 3.75
  - System inputs: For each type of system input flow, the receiving portion of the complexity is used: **0.5** = (1*3*0%: receive fuel) + (1*1*50%: receive ram air)
  - Internal links: For each type of internal flow, the full complexity is used: **64** = (3*4: electric power) + (1*1: pneumatic power) + (3*5: shaft power) + (7*2: heat) + (2*3: metered fuel) + (2*5: start shaft power) + (1*5: aux shaft power) + (1*1: aux pneumatic power)
  - System outputs: For each type of system output flow, the sending portion of the complexity is used: **3.75** = (1*4*50%: send electric power) + (1*1*0%: send cooling) + (1*0*100%: send flight control)
- Shared conveyance: 0 (no hierarchical clustering)
- Pass-thru: 0 (no hierarchical clustering)
- Complexity isolation: 0 (no hierarchical clustering)
- Clustering cost: 0 (no hierarchical clustering)

The result of applying the complexity metric to the architecture shown in Figure 15 is shown in Figure 16. Note that in Figure 16 the complexity metric values for each subsystem are shown in the lower right corner of each subsystem box, and that the overall complexity metric value for the system architecture is shown in the lower right corner of the overall system box.

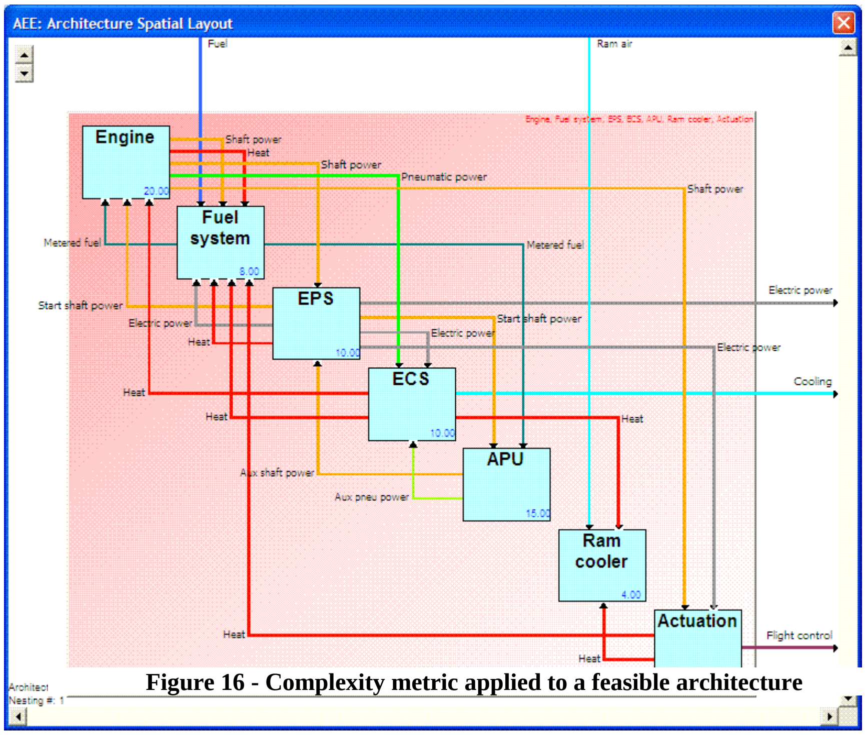


Figure 16 - Complexity metric applied to a feasible architecture

As described in [15], an architecture can be hierarchically clustered, using spectral graph partitioning methods coupled with complexity principles. Figure 17 shows a clustered version of the architecture shown in Figure 15 with a significantly lower complexity.

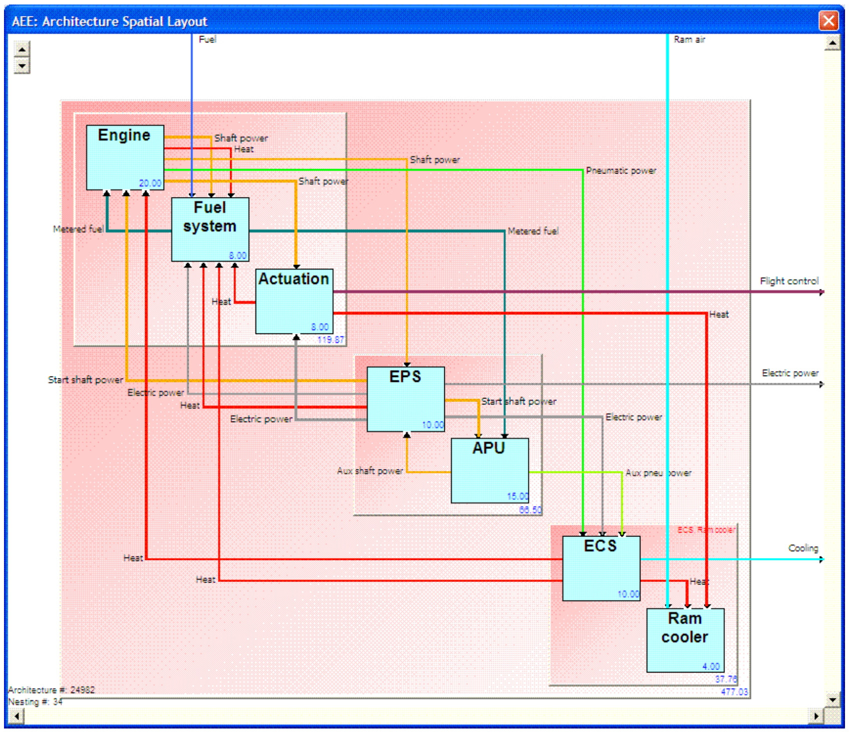


Figure 17 - Aerospace power & thermal management architecture clustered using BestWCut result.

V. Conclusion

A method for developing metrics that evaluate the complexity of large systems has been described. Such metrics are intended to be used early in the architecting of large systems as a proxy for their eventual total system cost, as an early means to make design decisions that reduce that total system cost. This method for developing metrics differs from prior methods in that it uses only the information available at the earliest stages of system architecting and represents a set of complexity principles, which can then be validated.

Acknowledgments

The development of the methods and results contained in this study were sponsored under the DARPA contract: “Abstraction Based Complexity Management” #FA9550-07-C-0024. [16]

References

[1]Zeidner, L.E., St. Rock, B.E., Desai, N.A., Reeve, H.M., and Strauss, M.P., “Application of a Technology Screening Methodology for Rotorcraft Alternative Power Systems,” *AIAA Aerospace Sciences Meeting*, AIAA-2010-1505, Orlando, FL 2010.

[2]Navlakha, J.K., “A Survey of System Complexity Metrics,” The Computer Journal, 1987, Vol. 30, No. 3, pp. 233-238.

[3]Meila, M. and Pentney, W., “Clustering by weighted cuts in directed graphs,” Proceedings of the 2007 SIAM International Conference on Data Mining, SIAM, pp. 135-144.

[4]McCabe, T.J., “A complexity measure. IEEE Transactions on Software Engineering,” 1976. Vol. 2, No. 4, pp. 308–320.

[5]Kim, J. and Wilemon, D. “An empirical investigation of complexity and its management in new product development”, Technology Analysis & Strategic Management, Vol. 21, Issue 4, 2009, pp. 547-564.

[6]Shepperd, M. and Ince, D.C., “A critique of three software metrics,” Journal of Systems and Software, 1994, Vol.26 , No. 3, pp. 197-210.

[7]Sangiovanni-Vincentelli, A., "Quo Vadis, SLD? Reasoning About The Trends and Challenges of System Level Design," Proceedings of the IEEE, 2007, Vol. 95, No. 3, pp. 467-506.

[8]Walters, E. A., and Iden, S., "INVENT Modeling, Simulation, Analysis and Optimization", AIAA Aerospace Sciences Meeting, Orlando, FL, 2010-287.

[9]Jones, R., Hardin, P., and Irvine, A., "Simple Parametric Model for Estimating Development (RDT&E) Cost", 2009 ISPA/SCEA Joint Conference.

[10]Arena, M. V., Younossi, O., Brancato, K., Blickstein, I., Grammich, C. A., "Why Has the Cost of Fixed-Wing Aircraft Risen?" RAND Corporation, 2008, http://www.rc.rand.org/pubs/monographs/2008/RAND_MG696.pdf.

[11]Histon, J. M., Hansman, R. J., Aigoin, G., Delahaye, D. & Puechmorel, S., "Introducing Structural Considerations into Complexity Metrics", 4th USA/Europe Air Traffic Management R&D Seminar (Orlando), 2001 (reproduced in ATC Quarterly, June 2002).

[12]Abbasi, M., "Perspectives of Complexity and Intelligence on Logistics and Supply Chain Management," Masters Thesis, 2008, University of Borås, Sweden.

[13]Pavard, B. and Dugdale, J., "The contribution of complexity theory to the study of socio-technical cooperative systems," New England Complex Institute electronic journal, 2000.

[14]Edmonds, B.. "Syntactic Measures of Complexity," Doctoral Thesis, University of Manchester, UK, 1999.

[15]Zeidner, L.E., Banaszuk, A., and Becz, S.B., "System Complexity Reduction via Spectral Graph Partitioning to Identify Hierarchical Modular Clusters," 10th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference, 2010.

[16]Becz, S., Pinto, A., Zeidner, L.E., Khire, R., Reeve, H.M., "Design System for Managing Complexity in Aerospace Systems", 10th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference, 2010.

F.7 System Complexity Reduction via Spectral Graph Partitioning to Identify Hierarchical Modular Clusters

System Complexity Reduction via Spectral Graph Partitioning to Identify Hierarchical Modular Clusters

Lawrence E. Zeidner[1], Andrzej Banaszuk[2] and Sandor B. Becz[1]
United Technologies Research Center, East Hartford, CT 06108, USA

The cost of large complex systems can be reduced by using system complexity as a cost proxy during the initial stages of system architecting. This requires evaluation of system complexity in terms of information available during those early stages. System modularity is a well-known technique for reducing system complexity by clustering system elements into modules to maximize connections within modules and minimize connections between them. This paper presents a method for identifying a hierarchical clustering of modules to reduce system complexity. This method employs spectral graph partitioning based on an adjacency matrix that is constructed using information available early in the design process, and specifically employs the WBestCut partitioning algorithm.

Nomenclature

DSM	=	Dependency Structure Matrix, Design Structure Matrix
GA	=	Genetic Algorithm
BestWCut	=	Best weighted cut algorithm [2]
A	=	Adjacency matrix
D_i	=	Node degrees (in-degree, out-degree or a combination)
T, T′	=	Diagonal Weighting Vectors
H(B)	=	Hermitian matrix

I. Introduction

Significant technological and architectural changes have been introduced into aerospace systems, over the past several decades, in an effort to improve the performance and capability of new platforms. These changes have led to an exponential growth in the complexity of modern aerospace platforms and accompanying design and development challenges. These increases in technical complexity have been accompanied by increases in the complexity of system requirements and organizational partnerships. Complexity in requirements stems from the need to meet multiple present and future mission requirements. Analysis by the RAND Corporation found that the largest component in the growth in the cost of fixed wing aircraft has come from increased complexity [44]. As systems have become more complex their development cost has increased and the predictability of their development cost has decreased. Managing and minimizing complexity of new system development offers the ability to reduce the magnitude and increase the predictability of development cost. Fundamental architectural decisions made early in the design process have a major impact on system complexity; however, system architects lack quantitative feedback on the complexity of their architectures. [43]

During the early design stages, large systems are architected in terms of functional subsystems (e.g., Electrical Power System, Environmental Control System, Auxiliary Power Unit) and their interconnections as shown in Figure 1- An aerospace system architecture specified in terms of its functional subsystems and their interconnections. This standard top-down design approach is a divide-and-conquer strategy that defines the sub-system boundaries as an interface to decouple the overall design problem into two sequential sub-problems. The first sub-problem is to design a system of functional subsystems based on overall system requirements, resulting in a set of requirements for each subsystem. The second sub-problem is to design each functional subsystem in terms of its required

[1] Research Engineer, Thermal Management, United Technologies Research Center, 411 Silver Lane, MS 129-89, East Hartford, CT, 06108.
[2] Fellow, Control Systems, United Technologies Research Center, 411 Silver Lane, MS 129-15, East Hartford, CT, 06108.

functionality. This approach has also enabled airframers to focus on the large-scale system-integration problem, while outsourcing subsystem design to subcontractors, with the subsystem requirements serving as the organizational interface. Since these two sub-problems are not really decoupled, subsystem interactions necessitate design iteration to meet system-level requirements. Because functional subsystems are designed separately, these iterations often occur late in the design process and are a significant cause of schedule and cost overruns.

One approach to design an overall system comprised of functional subsystems is to enumerate and evaluate all feasible solutions to this problem, as described in [1]. This approach has typically been avoided due to the exponential size of the design space. However, it has recently been shown that the feasible set is actually quite sparse, many orders of magnitude smaller than the overall design space, and that generative filters can be used to identify the entire set of feasible solutions directly, rather than searching through the design space for feasible solutions. [1] Using this approach, each architecture in the feasible set is specified in terms of the type of information depicted in Figure 1.

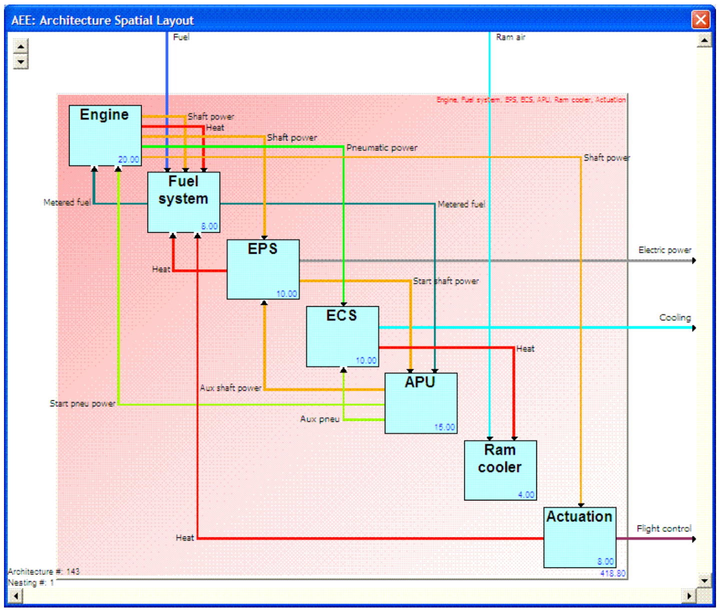


Figure 1- An aerospace system architecture specified in terms of its functional subsystems and their interconnections

The complexity of each architecture depends not only upon its constituent technologies and its interconnections, but also on how and to what degree the architecture is organized hierarchically into modules. Each architecture can be decomposed into many different hierarchical configurations of modules, each with its own degree of complexity. Complexity metrics suitable for hierarchical system configurations have been described in [41]. They roll up the complexity within each hierarchical module based on the complexity of its internal elements and interconnections, according to relevant principles describing complexity propagation and its relationship to cost.

This paper presents a method for determining the lowest-complexity hierarchical decomposition of a given architecture, based on a specified hierarchical complexity metric. This method uses a spectral graph partitioning algorithm iteratively to determine the hierarchy of modules based on the limited information available at this early stage in the design process.

Early in the design of an overall system comprised of functional subsystems, the subsystems themselves have not yet been designed or sized. The information available about each subsystem consists of the sub-system technology and its energy, fuel and/or data input and output flows. The information available about the interactions among the sub-systems, and between the subsystems and the system's environment, consists of the type of energy, fuel or data flowing through each interconnection. This information is used by the partitioning algorithm to determine the lowest-complexity hierarchical decomposition of the system architecture.

System modularity has been pursued in product design because it has been expected to lead to several desirable system benefits [3]:

- Improved quality
- Shorter development time
- Flexibility and variety
- Risk reduction
- Cost reduction.

A related design problem, the design of product families, which are sets of products that enjoy these benefits because they share common modules, has been an active field of study. Consequently, researchers have proposed various ways to cluster product designs into modules. However, some of these methods depend upon information that is not available in the early stages of system architecting described above:

- Flow-Distance Dendrograms: Holtta(14)
- Modular Function Deployment (MFD): Erixon(20)
- Interface Modeling: Andersson(19)

Other clustering methods are heuristics which require human judgement:

- DSM Relation (Heuristics): Blackenfelt(23)
- DSM Visual Inspection (Heuristics): Sosa(24)
- Visibility-Dependency (Heuristic): Yassine(10)
- Function Chains (Heuristics): Dahmus(27), Stone(13)
- Interaction-Graph (Heuristic): Kusiak(15)

Other clustering methods are based on searches of very large spaces with genetic algorithms, which lack stopping criteria, thus finding good but not necessarily optimal solutions, and which slow down for large combinatorial problems:

- GA Methods: Whitfield(4), Fujita(6), Whitfield(9), Kamrani(18)
- Minimal Description Length: Helmer(7), Yu (11), Wang (12)
- Multi-perspective DSM Distance-from-Diagonal: Meehan(8)

One set of clustering methods is graph partitioning which cuts a graph, consisting of nodes and edges connecting them, into two subgraphs to minimize the interconnections between the two subgraphs. Minimum cut methods simply minimize the number of edges connecting the two subgraphs. They tend to produce extremely imbalanced cuts that are of little value for high-level clustering. [35] Other methods impose an additional constraint to produce better-balanced cuts. Minimum bisection methods impose the constraint that the two subgraphs are of equal size. This is an unnatural constraint, yields poor clustering results quantitatively, and is NP-complete computationally. [34] Ratio cut methods minimize the ratio of the number of edges connecting the two subgraphs divided by the product of the number of nodes in each subgraph. [36, 37] They are a better choice for this application because they tend to produce naturally balanced and yet high-fitness cuts.

Spectral graph partitioning uses the eigenvalues and eigenvectors of an affinity matrix describing relationships between pairs of nodes in a graph [30-32] to find the cut, e.g., a ratio cut. For this application, the affinity matrix could be an adjacency matrix of the architecture's directed graph describing some properties of the directed interconnections between its subsystem nodes. The problem can be formulated as a Markov random walk along the edges of the graph, with the affinity matrix terms interpreted as the edge flows. [40]

Most spectral graph partitioning methods assume a symmetric affinity matrix, but in general, the adjacency matrix for a system architecture is asymmetric. Chung [29] explores the Laplacian of a directed graph, and Meila [2] presents a weighted-cut algorithm (BestWCut) that applies to asymmetric affinity matrices. BestWCut is parametrically tunable to adopt the behavior of a range of clustering methods. One of these is WNCut, which acts as a weighted normalized cut. This paper demonstrates applicability of this algorithm to hierarchical clustering of system architectures based on the data available early in the design process. An adjacency matrix, based on a system-complexity metric, is used as the affinity matrix. This algorithm is applied recursively to form the hierarchical clustering.

The remaining sections of this paper are organized as follows. Section II will define the problem, section III will present a solution approach, and section IV will provide an example application of this solution approach. The paper concludes with a summary and with suggested extensions.

II. Problem Definition

This paper presents a method for determining the lowest-complexity hierarchical decomposition of a given architecture, based on a specified hierarchical complexity metric. This method uses a spectral graph partitioning algorithm recursively to determine the hierarchy of modules based on the limited information available at this early stage in the design process.

The focus of this paper is the spectral graph partitioning; however, to understand the information available and how the result is used, the following subsections will present the context in which the spectral graph partitioning problem is solved. The partitioning method is one of three techniques that are employed together to reduce the enormous design space to a manageable size before proceeding to a lower level of abstraction, using higher fidelity models: architectural enumeration, a complexity metric, and spectral graph partitioning for hierarchical system clustering.

A. Architectural Enumeration

The entire architectural design space is considered rapidly to identify the feasible set of architectures that meet configuration requirements. [1] The architectural design space is enormous, but the feasible set is extremely sparse. Generative filters are used to rapidly consider the entire design space and identify the entire feasible set. The architectural design space is specified by identifying the technology alternatives at the relevant level (e.g., device, subsystem), the energy/fuel/data flow types between devices or subsystems, and the energy/fuel/data inputs and outputs of each technology alternative. Architectural enumeration results in a list of feasible architectures.

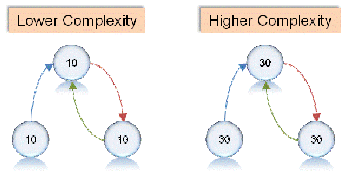
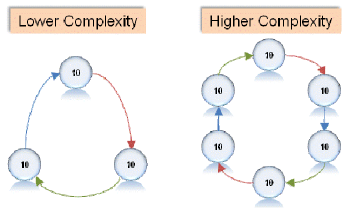
Architectural enumeration begins with the selection of relevant component technologies, the identification of compatible input and output energy flow types, and the identification of application requirements and constraints as filters. The Knowledge Center is an organizational repository of component technology descriptions.

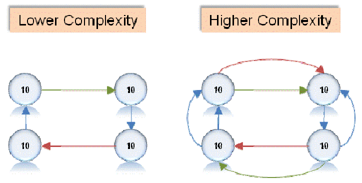
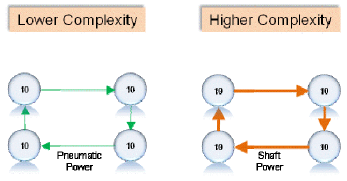
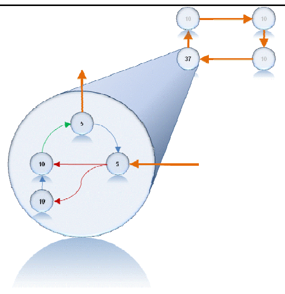
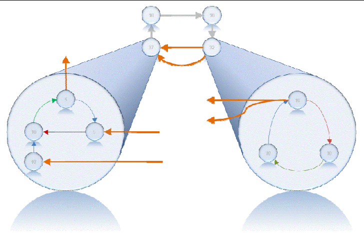
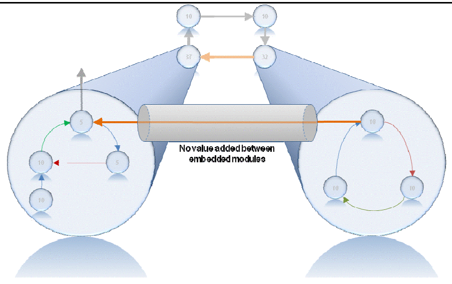
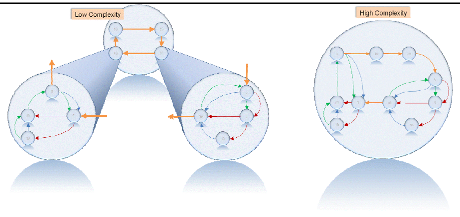
Once the inputs and filters have been specified, architectural enumeration performs rapid design-space reduction according to those inputs and filters. The exponential size of the design space necessitates an approach that doesn’t implement the filter criteria as “evaluative” filters, constructing each point in the design space and then evaluating it. Instead, “generative” filters use knowledge of the problem structure to directly identify the points in the feasible set, with minimal computational overhead. [1]

B. Complexity Metric

A complexity metric can be created to represent principles regarding the propagation of complexity and how it drives system cost. [41] Table 1 provides several examples of complexity principles believed to be relevant to aerospace systems; however, the purpose of including them in this paper is not to focus on the specific principles themselves, but to illustrate how such principles can be combined to create a complexity metric that embodies them. This background will help the reader to understand how the complexity metric is implemented in hierarchically iterative execution of spectral graph partitioning.

Table 1 - Complexity Principles

Principle	Description	Diagram
Node complexity	A system’s complexity increases as the complexity of its constituent parts increases.	 Lower Complexity 10 10 10 Higher Complexity 30 30 30
Number of nodes	The complexity increases as the number of constituent parts increases.	 Lower Complexity 10 10 10 Higher Complexity 10 10 10 10 10 10

Number of links	The complexity increases as the number of interconnections increases.	
Types of links	The complexity that each type of interconnection adds to a system is based on the type of energy/fuel/data flow.	 Lower Complexity Higher Complexity 10 Pneumatic Power Shaft Power
Roll-up	A module encapsulates the complexity of the elements inside it and their interconnections with one another and with the module's environment.	
Shared conveyance	Multiple interconnections of the same energy/fuel/data flow type linking two embedded modules may present lower complexity together than separately.	
Pass-through	An interconnection passing from one embedded subsystem to another represents complexity within those embedded systems, but little complexity in their parent system, which adds minimal value to the flow.	
Complexity isolation	Clustering nodes and their interconnections into hierarchical modular clusters localizes complexity so that other modules and higher levels may have lower complexity.	

Clustering cost	While clustering is beneficial, there are design, fabrication, test, maintenance and other costs associated with each modular interface. The benefit of complexity isolation is balanced in the complexity metric by an associated cost.	

Complexity principles can be embedded in a complexity metric. For example, Jones, et al. [42] present a complexity metric that embodies a few of these principles: node complexity, number of nodes, and number of links. Their complexity metric is:

$$RDT\&E\$ = aN_e^b \quad \text{where}$$

$$N_e = \underbrace{\left(aN_{s/r} + bN_s + cN_r\right)}_{\text{Capture complexity associated with type of Nodes}} \underbrace{\left(\frac{L_t/N_t}{\text{avg}\left(L_t/N_t\right)}\right)^d}_{\text{Capture the connectivity complexity of the system}}$$

The node complexity principle is embodied in the distinction between three types of nodes: receivers (N_r), transmitters (N_s), and transceivers ($N_{s/r}$), and their multiplication by three separate complexity factors, h, g and d. The number of nodes principle is embodied in multiplication by the term ($dN_{s/r} + gN_s + hN_r$). The number of links principle is embodied by the term L_t.

C. Spectral Graph Partitioning for Hierarchical System Clustering

Spectral methods are applied iteratively to each architecture to cut the architecture into hierarchical modular clusters that minimize the complexity of that architecture as measured by the complexity metric. The partitioning method enables principles of cost due to complexity to be represented in the graph adjacency matrix used as input. The adjacency matrix describes the energy/fuel/data flows between devices or subsystems. Principles regarding the propagation of complexity and how it drives cost are embodied in the way those flows are quantified, in the adjacency matrix, as the spectral methods are applied iteratively to create the hierarchy. These spectral methods are the subject of this paper.

The technical challenge is to partitioning a system architecture rapidly into a hierarchical set of modules that will minimize system complexity as measured by a complexity metric. The system architecture is expressed as a graph of subsystems interconnected by edges along which energy, fuel or data flow. This graph is a multidigraph, a directed graph in which each pair of vertices can have multiple edges connecting them.

III. Solution Approach

This section will describe the spectral graph partitioning solution approach applied iteratively to hierarchically cluster the subsystems. The solution approach is based on iterative application of the BestWCut algorithm presented in Meila[2]. In addition, as BestWCut is focused on different levels within the architecture, certain cross-cutting complexity principles will be applied, as part of the complexity metric, to form the adjacency matrix. The adjacency matrix serves as an interface between the BestWCut algorithm and both the architectural enumeration and complexity metric.

The BestWCut algorithm [2] is:

```
Input Affinity matrix A, weights T, T′, number of clusters K.
    1. A ← T′A
    2. $D_i \leftarrow \sum_{j=1}^{n} A_{ij}$ , D = diag{$D_i$}$_i$
```

3. $H(B) \leftarrow \frac{1}{2} T^{-\frac{1}{2}} (2D - A - A^T) T^{-\frac{1}{2}}$
4. Compute Y the n × K matrix with orthonormal columns containing the K smallest eigenvectors of H(B)
5. Cluster the rows of $X = T^{-\frac{1}{2}}Y$ as points in $\mathbf{R}^K$.
   (**Variant**: Normalize the rows of Y to have length 1, then cluster them as points in $\mathbf{R}^K$.)

The affinity or adjacency matrix A has the distinction, among spectral graph partitioning algorithms, of being asymmetric, which is necessary to represent a directed multigraph. However, it is useful to modify step 2 in the algorithm so that the D_i are a function of both the in-degree and the out-degree of the graph vertices. Meila [2] identifies this definition of D_i as a "direct authority" model, in which a node's importance is equal to the number of its outgoing links. Meila suggests transposing A to address problems that are dominantly in-bound, such as citations and internet web links, however, for a general directed multigraph, the forward and backward links, above and below the diagonal, are of equal importance. A suitable alternative for task 2 is to define D_i as the average of the in-degree and out-degree, although other formulations may work as well:

$$D_i \leftarrow \left(\left(\sum\nolimits^{n}_{i=1} A_{ij}\right) + \left(\sum\nolimits^{n}_{j=1} A_{ij}\right) \right) /2.$$

In addition, with this definition for the D_i, the clustering in step 5 achieves better results if it algorithmically ignores any constant eigenvectors.
The next section will explain how the WBestCut algorithm can be applied hierarchically to cluster a system.

IV. Example Application

The example application is drawn from aircraft design and focuses on power and thermal management. The system is described in terms of a set of subsystems and their interconnections via energy, fuel and data flows, including flows to and from the rest of the aircraft.

A. Architectural Enumeration

There are seven subsystems, which are listed in Table 2

Table 2 - Subsystems, their functions and complexity values

Subsystem	Function	Complexity
Engine	the focus here is on power for the other subsystems (not thrust)	20
Fuel System	meters fuel	8
Electric Power System	provides electric power for subsystems and aircraft hotel loads	10
Environmental Control System	provides aircraft cooling	10
Auxiliary Power Unit	provides auxiliary power when the Engines are not available	15
Ram Cooler	provides a heat sink	4
Actuation	provides flight control	8

There are 15 types of energy, fuel and data flows (listed in Table 3 in order of decreasing complexity):

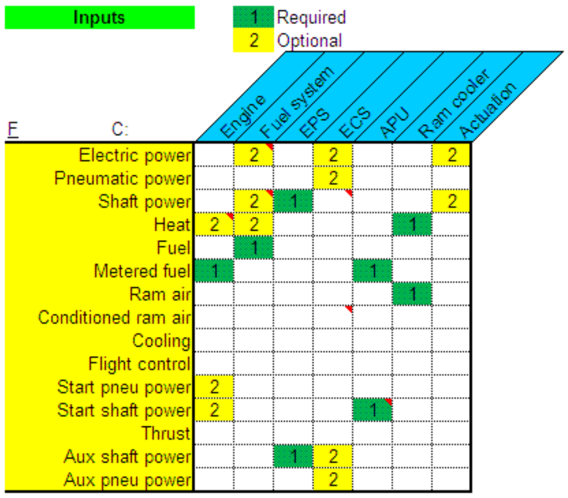
Table 3 - Inputs: Energy, fuel and data flows

Energy, fuel & data flows					Complexity			
Name	Power	Available System Inputs	Desired Output	Link color	Total	Send	Convey	Receive
Electric power	1	0	1	▨	4	50%	0%	50%
Pneumatic power	1	0	0	🟩	1	50%	0%	50%
Shaft power	1	0	0	🟧	5	33%	33%	33%
Heat	0	0	0	🟥	2	33%	33%	33%
Fuel	0	1	0	🟦	3	50%	50%	0%
Metered fuel	0	0	0	▨	3	50%	50%	0%
Ram air	0	1	0	🟦	1	50%	0%	50%
Conditioned ram air	0	0	0	🟦	1	50%	0%	50%
Cooling	0	0	1	🟦	1	75%	25%	0%
Flight control	0	0	1	🟪	0	100%	0%	0%
Start pneu power	1	0	0	🟩	1	50%	0%	50%
Start shaft power	1	0	0	🟧	5	33%	33%	33%
Thrust	1	0	0	▨	0	100%	0%	0%
Aux shaft power	1	0	0	🟧	5	33%	33%	33%
Aux pneu power	1	0	0	🟩	1	50%	0%	50%

Each flow has a name, may be a form of power, may be an available system input or a desired system output, has a link color (for display on the architectural layout), has a total complexity (according to Link types complexity principle), and has a breakdown of that total complexity into the complexity to send, convey and receive that type of flow.

Each type of subsystem can accept certain types of energy, fuel and data flows (listed in Table 4):

Table 4 - Subsystem flow inputs



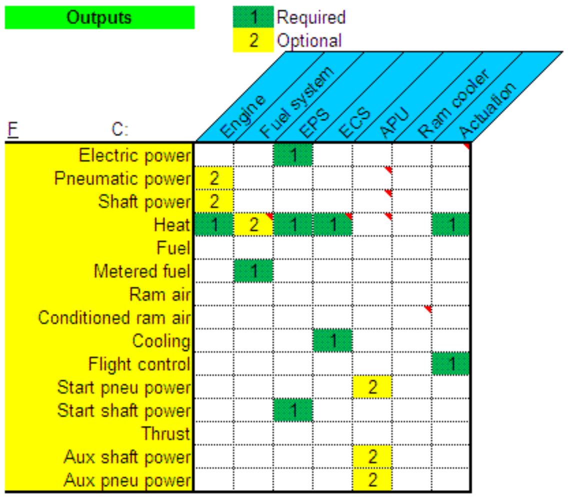
Inputs

1	Required
2	Optional

F C:	Engine	Fuel system	EPS	ECS	APU	Ram cooler	Actuation
Electric power		2		2			2
Pneumatic power				2			
Shaft power		2	1				2
Heat	2	2				1	
Fuel		1					
Metered fuel	1				1		
Ram air						1	
Conditioned ram air							
Cooling							
Flight control							
Start pneu power	2						
Start shaft power	2				1		
Thrust							
Aux shaft power			1	2			
Aux pneu power				2			

Likewise, each type of subsystem can produce certain types of energy, fuel and data flows (listed in Table 5):

Table 5 - Subsystem flow outputs



Outputs

1	Required
2	Optional

F C:	Engine	Fuel system	EPS	ECS	APU	Ram cooler	Actuation
Electric power			1				
Pneumatic power	2						
Shaft power	2						
Heat	1	2	1	1			1
Fuel							
Metered fuel		1					
Ram air							
Conditioned ram air							
Cooling				1			
Flight control							1
Start pneu power					2		
Start shaft power			1				
Thrust							
Aux shaft power					2		
Aux pneu power					2		

The architectural enumeration for this problem produced ~20,000 feasible architectures in under 1 minute. These architectures pass all of the specified requirements and constraints, so they are feasible in the sense that they are configured feasibly. At this point in the process, neither complexity nor other value metrics such as performance, weight, durability, etc. have been evaluated.

One of the feasible architectures is shown in Figure 2. It will be examined further below.

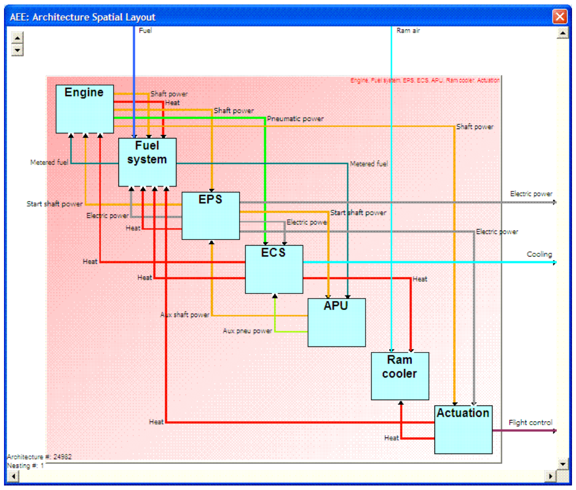


Figure 2 - A feasible architecture

B. Complexity Metric

Following the WBestCut algorithm (Section III Solution Approach), we begin with the inputs. The diagonal weights T and T' are both set to 1 to emulate an average cut, so T = T' = I (identity matrix).

The adjacency matrix is built using the complexity principles represented in the complexity metric. For each pair of nodes in the architecture (**Error! Reference source not found.**) i and j, the adjacency matrix term Aij is the sum of the complexities of the directed links from node i to node j. The <u>types of links</u> principle is applied to determine the link complexities associated with each interconnecting energy / fuel / data flow (Table 6):

Table 6 - <u>Types of links</u> and complexities.

	Complexity
Shaft power	5
Start shaft power	5
Aux shaft power	5
Electric power	4
Fuel	3
Metered fuel	3
Heat	2
Pneumatic power	1
Ram air	1
Conditioned ram air	1
Cooling	1
Start pneu power	1
Aux pneu power	1

The resulting adjacency matrix is shown in Table 7:

Table 7 - Adjacency matrix.

	Engine	Fuel Sys	EPS	ECS	APU	Ram Cooler	Actuation
Engine		🟨🟥 7	🟨 5	🟩 1			🟨 5
Fuel Sys	🟦 3				🟦 3		
EPS	🟨 5	⬜🟥 6		4	🟨 5		⬜ 4
ECS	🟥 2	🟥 2				🟥 2	
APU			🟨 5	🟩 1			
Ram Cooler							
Actuation		🟥 2				🟥 2	

The adjacency matrix is then normalized to sum to 1, resulting in the adjacency matrix A, shown in Table 8:

Table 8 - Normalized adjacency matrix.

	Engine	Fuel Sys	EPS	ECS	APU	Ram Cooler	Actuation
Engine		0.109	0.078	0.016			0.078
Fuel Sys	0.047				0.047		
EPS	0.078	0.094		0.063	0.078		0.063
ECS	0.031	0.031				0.031	
APU			0.078	0.016			
Ram Cooler							
Actuation		0.031				0.031	

C. Spectral Graph Partitioning

This section will show how the BestWCut algorithm is applied, given this normalized adjacency matrix which embodies the relevant complexity principles. The D matrix is constructed so that the D_i terms are the average of the ith row sum and the ith column sum of A, to include both outgoing and incoming link complexity of each node, as shown in Table 9:

Table 9 - D matrix.

	Engine	Fuel Sys	EPS	ECS	APU	Ram Cooler	Actuation
Engine	0.219						
Fuel Sys		0.180					
EPS			0.266				
ECS				0.094			
APU					0.109		
Ram Cooler						0.031	
Actuation							0.102

The Hermitian matrix H(B) is defined in step 3 of the WBestCut algorithm as:

3. $H(B) \leftarrow \frac{1}{2} T^{-\frac{1}{2}} (2D - A - A^T) T^{-\frac{1}{2}}$

However, since T is an identity matrix, we can simplify to:

$H(B) \leftarrow \frac{1}{2} (2D - A - A^T)$

So the resulting Hermitian matrix is shown in Table 10:

Table 10 - Hermitian matrix H(B).

	Engine	Fuel Sys	EPS	ECS	APU	Ram Cooler	Actuation
Engine	0.219	-0.078	-0.078	-0.023			-0.039
Fuel Sys	-0.078	0.180	-0.047	-0.016	-0.023		-0.016
EPS	-0.078	-0.047	0.266	-0.031	-0.078		-0.031
ECS	-0.023	-0.016	-0.031	0.094	-0.008	-0.016	
APU		-0.023	-0.078	-0.008	0.109		
Ram Cooler				-0.016		0.031	-0.016
Actuation	-0.039	-0.016	-0.031			-0.016	0.102

The eigenvalues and eigenvectors of the Hermitian matrix are shown in Table 11:

Table 11- Eigenvalues and Eigenvectors of H(B).

Eigenvalues:		
	0.000	1
	0.032	2
	0.094	3
	0.106	4
	0.157	5
	0.268	6
	0.343	7

Eigenvectors:	1	2	3	4	5	6	7
	0.378	0.179	0.153	0.130	0.399	0.587	0.531
	0.378	0.205	0.067	0.023	0.597	-0.670	-0.071
	0.378	0.197	-0.030	-0.107	0.063	0.407	-0.798
	0.378	0.025	-0.550	0.671	-0.307	-0.089	0.046
	0.378	0.265	-0.336	-0.687	-0.343	-0.097	0.272
	0.378	-0.903	-0.048	-0.171	0.101	0.016	-0.003
	0.378	0.032	0.744	0.141	-0.509	-0.153	0.022

Since the first eigenvector is constant, it is discarded. For k=2, the 2nd and 3rd eigenvectors are clustered in R^2 yielding two clusters, as shown in Table 12:

Table 12 - Clusters for K=2.

	Cluster
Engine	1
Fuel Sys	1
EPS	1
ECS	2
APU	1
Ram Cooler	2
Actuation	1

The architectural layout corresponding to this clustering is shown in Figure 3:

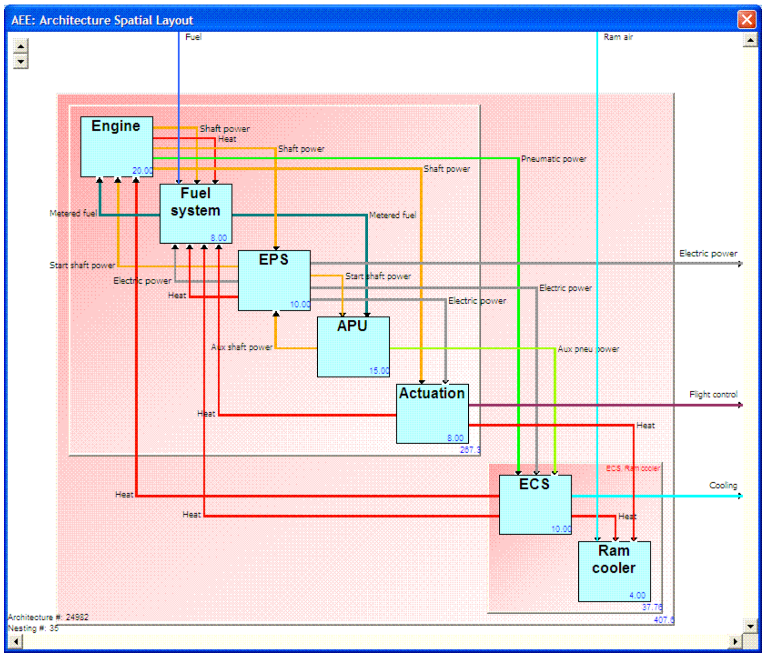


Figure 3 - Architecture clustered with K=2.

The architecture clustered with K=3 is shown in Figure 4:

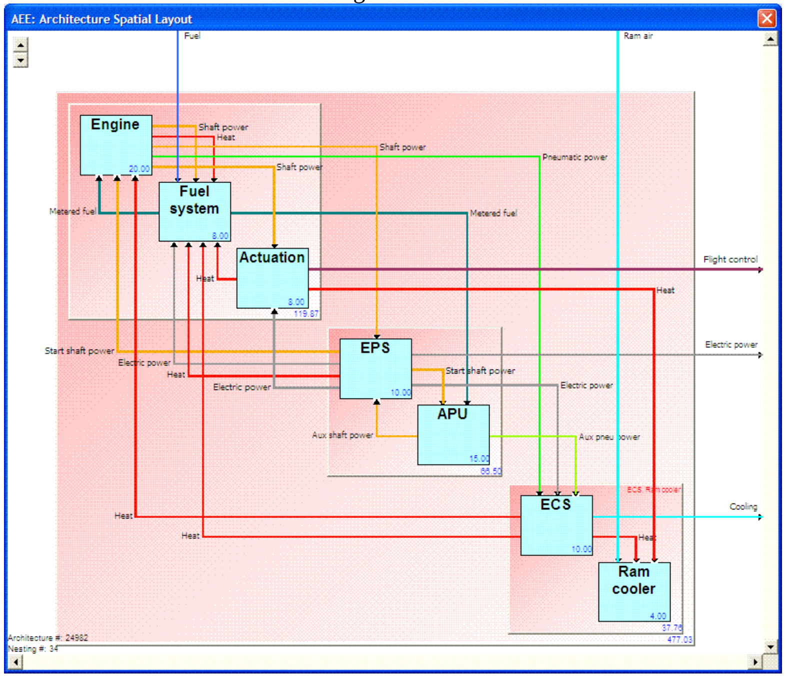


Figure 4 - Architecture clustered with K=3.

V. Conclusion

A method for minimizing the complexity of large systems by hierarchically clustering them and thereby reducing their total system cost has been described. It differs from prior methods in that it uses only the information available at the earliest stages of system architecting, is fully automated, and is computationally faster than search algorithms.

Acknowledgments

The development of the methods and results contained in this study were sponsored under the DARPA contract: "Abstraction Based Complexity Management" #FA9550-07-C-0024. [43]

References

[1]Zeidner, L.E., St. Rock, B.E., Desai, N.A., Reeve, H.M., and Strauss, M.P., "Application of a Technology Screening Methodology for Rotorcraft Alternative Power Systems," *AIAA Aerospace Sciences Meeting*, AIAA-2010-1505, Orlando, FL 2010.

[2]Sahai, T., Speranzon, A. and Banaszuk, A., "Hearing the clusters in a graph: A distributed algorithm," *Proceedings of the National Academy of Sciences of the United States of America*, 2010.

[3]Meila, M. and Pentney, W., "Clustering by weighted cuts in directed graphs," Proceedings of the 2007 SIAM International Conference on Data Mining, SIAM, pp. 135-144.

[4]Whitfield, R.I., Smith, J.S., and Duffy, A.H.B., "Identifying Component Modules," 7th International Conference on Artificial Intelligence in Design (AID '02), Cambridge, UK, pp. 571-592, ISBN: 1-4020-0716-7.

[5]Ishii, K. and Yang, T.G., "Modularity: International Industry Benchmarking and Research Roadmap," Proceedings of DETC'03 ASME 2003 Design Engineering Technical Conference and Computers and Information in Engineering Conference, Chicago, Illinois, 2003, DETC2003/DFM-48132.

[6]Fujita, K and Yoshida, H, "Product Variety Optimization: Simultaneous Optimization of Module Combination and Module Attributes," Proceedings of DETC'01 ASME 2001 Design Engineering Technical Conference and Computers and Information in Engineering Conference, Pittsburg, PA, 2001, DETC2001/DAC-21058.

[7]Helmer, R. Yassine, A. and Maier, C., "Module and Interface Identification and Definition – A Comprehensive Approach Using DSM's," 9th International Conference on Engineering Design, ICED'07.

[8]Meehan, J.S., Duffy, A.H.B., and Whitfield, R.I., “Supporting ‘Design for Re-use’ with Modular Design,” Concurrent Engineering: Research and Applications, Vol. 15, No. 2, 2007, pp. 141-155.

[9]Yan, X.T., Stewart, B., Wang, W., Tramsheck, R., Liggat, J., Duffy, A.H.B., and Whitfield, I, “Developing and Applying an Integrated Modular Design Methodology within a SME,” International Conference on Engineering Design, ICED’07/250.

[10]Sharman, D.M. and Yassine, A.A., “Characterizing Complex Architectures,” Systems Engineering, Vol. 7, No. 1, 2004, pp. 35-60.

[11]Yu, T.L., Yassine, A.A., and Goldberg, D.E., “An information theoretic method for developing modular architectures using genetic algorithms,” Res Eng Design, 2007, Vol. 18, pp. 91-109.

[12]Wang, B. and Antonsson, E.K., “Hierarchical Modularity: Decomposition of Function Structures with the Minimal Description Length Principle,” Proceedings of DETC’05, 2005 ASME Design Engineering Technical Conferences, Long Beach, CA, DETC2005/DTM-85173.

[13]Stone, R.B., Wood, K.L., and Crawford, R.H., “A Heuristic Method for Identifying Modules for Product Architectures,” Design Studies, Vol. 21, No. 1, 2000, pp. 5-31.

[14]Holtta, K., Tang, V., and Seering, W. P., “Modularizing Product Architectures Using Dendrograms,” International conference on engineering design, ICED03. Stockholm. 2003.

[15]Kusiak, A., Huang, C., “Development of Modular Products,” IEEE Transactions on Components, Packaging, and Manufacturing Technology – Part A, Vol. 19, No. 4, 1996, pp. 523-538.

[16]Miller, T.D., and Elgard, P., “Defining modules, modularity and modularisation,” Design for Integration in Manufacturing, Proceedings of the 13th IPS Research Seminar, Fugsloe, 1998.

[17]Fleming, L., and Sorenson, Olav, “The Dangers of Modularity,” Harvard Business Review, September 2001, pp. 20-21.

[18]Kamrani, A.K., and Gonzalez, R., “A genetic algorithm-based solution methodology for modular design,” Journal of Intelligent Manufacturing, Vol. 14, 2003, pp. 599-616.

[19]Andersson, S. and Sellgren, U., "Modular product development with a focus on modelling and simulation of interfaces," Design Society – Workshop on Product Structuring, Copenhagen, 2003.

[20]Erixon, G., “Modular Function Deployment (MFD), Support for Good Product Structure Creation”, Presented at the 2nd WDK Workshop on Product Structuring, Delft, Holland, 1996.

[21]Whitney, D.E., “Physical Limits to Modularity”, Working paper, ESD-WP-2003-01.03-ESD, MIT, Engineering Systems Division, 2004.

[22]Fixson, S.K., “Modularity and Commonality Research: Past Developments and Future Opportunities,” Concurrent Engineering Research and Applications, 2007, Vol. 15, No. 2, pp. 85-112.

[23]Larses O. & Blackenfelt M. 2003. “Relational reasoning supported by quantitative methods for product modularization,” Proceedings 14th International Conference on Engineering Design. ICED 03. Stockholm, 2003.

[24]Sosa, M.E., Eppinger, S.D., Rowles, C.M., “Identifying Modular and Integrative Systems and Their Impact on Design Team Interactions,” Transactions of the ASME, Vol. 125, 2003, pp. 240-252.

[25]Erens, F.J., and Verhulst, K., “Architectures for Product Families, “ WDK Workshop on Product Structuring, Delft University of Technology, 1996.

[26]Ward, J.H. Jr., “Hierarchical Grouping to Optimize an Objective Function,” Journal of the American Statistical Association, Vol. 58, Issue 301, 1963, pp. 236-244.

[27]Dahmus, J.B., Gonzalez-Zugasti, J.P., Otto, K.N., “Modular Product Architecture,” Proceedings of DETC’00: Design Engineering Technical Conferences and Computers and Information in Engineering Conference, DETC2000/DTM-14565.

[28]Meila, M. and Shi, J., “A random walks view of spectral segmentation,” T. Jaakkola and T. Richardson, editors, Artificial Intelligence and Statistics AISTATS, 2001.

[29]Chung, F., “Laplacian and the Cheeger inequality for directed graphs,” Annals of Combinatorics, Vol. 9, 2005, 1-19.

[30]Fiedler, M., “Algebraic connectivity of graphs,” Czechoslovak Mathematical Journal, Vol. 23, 1973, pp. 289–305.

[31]Fiedler, M., “A property of eigenvectors of nonnegative symmetric matrices and its application to graph theory,” Czechoslovak Mathematical Journal, Vol. 25, No. 4, 1975, pp. 619–633.

[32]von Luxburg, U., “A tutorial on spectral clustering,” Statistics and Computing, Vol. 17, No. 4, 2007, pp. 395–416.

[33]Ford, L.R., Jr. and Fulkerson, D.R., *Flows in Networks*, Princeton University Press, Princeton, NJ, 1962.

[34]Charney, H.R., and Plato, D.L., “Efficient partitioning of components,” Proc. IEEE Design Automation Workshop, 1968, pp. 16-0 – 16-21.

[35]Stoer, M. and Wagner, F., “A simple min-cut algorithm,” Journal of the ACM, 44(4):585–591, 1997.

[36]Wei, Y.C., and Cheng, C.K., “Towards efficient hierarchical designs by ratio cut partitioning,” Proc. IEEE Int. Conf. on Computer-Aided Design, 1989, pp. 298-301.

[37]Leighton T. and Rao, S., “An approximate max-flow min-cut theorem for uniform multicommodity flow problems with applications to approximation algorithms,” Proc. IEEE Ann. Symp. Found. Computer Sci., 1988, pp. 422-431.

[38]Donath, W.E., and Hoffman, A.J., “Algorithms for partitioning of graphs and computer logic based on eigenvectors of connection matrices,” IBM Technical Disclosure Bulletin, Vol. 15, 1972, pp. 938-944.

[39]Pentney, W. and Meila, M., “Spectral clustering of biological sequence data,” Proceedings of Twentieth National Conference on Articial Intelligence (AAAI-05), 2005.

[40]Meila, M. and Shi, J., “A random walks view of spectral segmentation,” Artificial Intelligence and Statistics AISTATS, 2001.

[41]Zeidner, L.E., Becz, S.B., Reeve, H.M., and Khire, R., "Design Issues for a Bottom-Up Complexity Metric Applied to Hierarchical Systems," 10th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference, 2010

[42]Jones, R., Hardin, P., and Irvine, A., "Simple Parametric Model for Estimating Development (RDT&E) Cost", 2009 ISPA/SCEA Joint Conference.

[43]Becz, S., Pinto, A., Zeidner, L.E., Khire, R., Reeve, H.M., "Design System for Managing Complexity in Aerospace Systems", 10th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference, 2010.

[44]Arena, M. V., Younossi, O., Brancato, K., Blickstein, I., Grammich, C. A., "Why Has the Cost of Fixed-Wing Aircraft Risen?" RAND Corporation, 2008, http://www.rc.rand.org/pubs/monographs/2008/RAND_MG696.pdf.